Theoretical Computer Science 410 (2009) 1486–1494

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Perfectly quilted rectangular snake tilings

Robert Brijder, Hendrik Jan Hoogeboom *

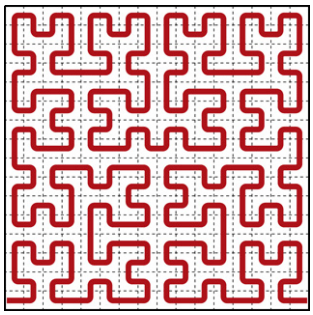
Leiden Institute of Advanced Computer Science, Universiteit Leiden, Leiden, The Netherlands

ARTICLE INFO

Keywords:
Picture languages
Tiling systems
Iterated substitutions

ABSTRACT

We introduce a particular form of snake tilings to define picture languages, and relate the obtained family to the recognizable picture languages (as defined by Wang tiles). The correspondence for substitution tilings is even closer, and hence is applicable to the Hilbert curve.



1. Introduction

We consider languages consisting of pictures [8], which here are rectangular arrays of symbols, assembled from unit-sized squares carrying one symbol each. The squares should fit together by their edges, like in a jig-saw puzzle. However, one abstracts from the familiar curved form of the edges in a jig-saw puzzle by instead considering marked edges, requiring that adjacent squares in the assembled rectangle have matching markings on their common edge.

Inspiration for this research is the well-known Hilbert curve as represented in [12], depicted, slightly simplified, above. An old decidability problem on the formation of infinite snake tilings could be solved [1] using an intricate tiling system devised earlier by Kari in the context of cellular automata. The system is built from the usual four-sided tiles, but it is applied to *snake tilings*, where the tiles are connected as a meandering ribbon, each tile matching only two adjacent tiles (rather than four). So, although the picture of the Hilbert tiling above looks like a snake, a continuous line visiting each unit of the resulting square once, this is not the underlying (four-sided) tiling as defined by Kari.

In this paper we take the intuitive image of a snake that covers all positions of a rectangle as our definition. Snake tilings basically are one-dimensional descriptions of pictures. Each such description consists of a single string built from the labels of the squares visited and the directions (up, right, down, left) in which the description continues. They can be 'programmed'

* Corresponding author.
E-mail address: hoogeboo@liacs.nl (H.J. Hoogeboom).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.12.010

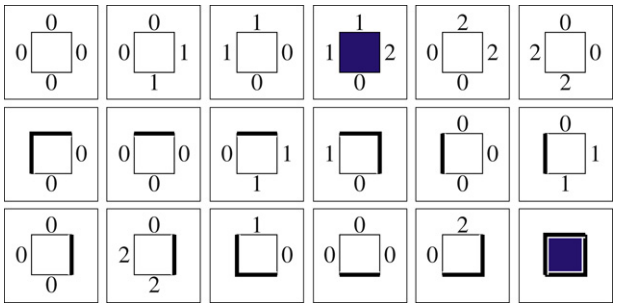


Fig. 1. Four-sided tiles that form a square with blue centre, see Example 1. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

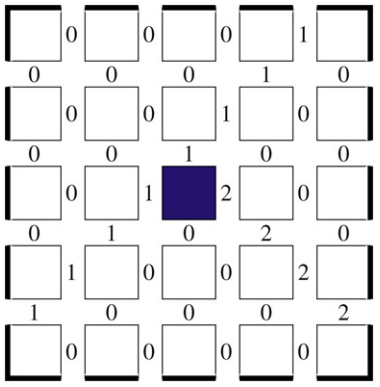


Fig. 2. Tiled rectangle using the tiles from Fig. 1. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

very much like automata walking over the grid, passing through positions of the picture, and checking their contents. Such a 'program' is then a regular language, states of an automaton corresponding to the markings on the tiles.

We compare the descriptional power of these regular snake tilings with the classic notion of finite tiling systems. We will show that, up to a multiplication factor, the requirement that the snake covers all positions of a rectangle exactly once makes them equivalent to tilings with four-sided tiles, the recognizable picture languages.

2. Definitions

Let $[m] = \{1, \ldots, m\}$. We start with a classic tiling model, which is sometimes known under the name of *Wang tiles*, except that here we are interested in tiling finite rectangles, rather than in tiling the plane. A *tiling* system is a finite set of (oriented) unit squares T each having its four edges marked by symbols from an alphabet Σ, or by the designated marking $\sharp \notin \Sigma$. A *T-tiling* of a $m \times n$ rectangle is a mapping from $[m] \times [n]$ into T such that adjacent tiles share the marking of their common edge. Moreover we require that the outer border of the rectangle is completely marked by $\sharp$, and that $\sharp$ does not occur internally. This enables us to fix some properties of the border tiles. Given a tile labelling (*colouring*) $\varphi : T \to \Delta$, the *picture language* defined by (Δ, T, φ) is the set of Δ-labelled rectangles that admit a tiling, i.e., $\{\varphi \circ t \mid t \text{ is } T\text{-tiling of a rectangle}\}$.

Alternatively, tilings can be defined without edge markings, instead restricting the allowed 2×2 tile patterns (and considering φ as we do here). The picture languages obtained in this way as 'projections of local languages' are often called the *recognizable* picture languages, see [8]. It is not hard to see that this definition is equivalent to the one with marked edges we use, see [15] for a formal proof.

Here we adhere to this terminology, and we call the picture languages defined by Wang tiles (a tiling system with a tile colouring) recognizable.

Example 1. Consider the tiling system consisting of 18 tiles as depicted in Fig. 1. The (internal) edges are marked by $\{0, 1, 2\}$ as indicated in the picture, where the special border marking $\sharp$ is replaced by a thick line. The colour of the squares (white or blue) indicates the value of the mapping φ.

The edge markings 1 and 2 start in a corner of the rectangle, and lie along a diagonal. In this way these tiles are forced to form a square of odd side length, with the blue tile taking the middle position (here the markings 1 and 2 meet). The tile labelling φ 'removes' the edge markings, leaving squares with a single coloured unit square in the middle. An example of a tiled rectangle (a 5×5 square) is given in Fig. 2.

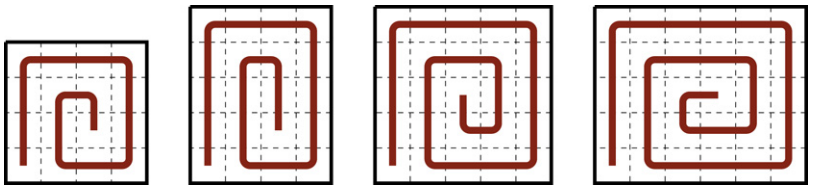


Fig. 3. Finding the middle, spiralling inward.

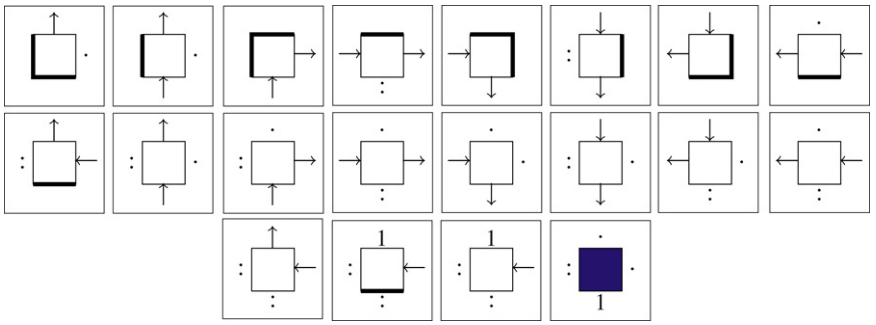


Fig. 4. The set of tiles used for the snake tilings in Example 2, implementing snakes as in Fig. 3 (third diagram). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

If we add for each white tile a blue copy, then we obtain the language of square white and blue tiles with blue middle. Note that this recognizable picture language cannot be accepted by *deterministic* automata walking over grids [3]. □

Snakes. The new model we propose here is based on the notion of a snake, as it occurs in the theory of tile connectivity problems, see [5] for a fundamental study (including an overview of the fascinating history of tiling problems). Here the snakes we consider will cover (all positions of) rectangles.

A *T-snake* on a $m \times n$ rectangle is a pair of mappings $\mu : [k] \to [m] \times [n]$ and $\nu : [k] \to T$, such that μ is injective, consecutive positions $\mu(i)$ and $\mu(i+1)$ are adjacent in the rectangle, the assigned tiles $\nu(i)$ and $\nu(i+1)$ share the marking at their common edge, and the borders of the rectangle are marked by $\sharp$. Tiles in the snake may touch at other places, but there we do not force matching markings (sometimes this is called a *weak* snake). A snake is called *perfectly quilted* if μ is bijective (so $k = mn$). In a sense, the mapping μ then describes a walk over the $m \times n$ rectangle, stepping from a tile to one of its four neighbours, where each position is visited exactly once. The tiles have matching edge markings along the walk described by the snake. Given a tile labelling φ as above, a perfectly quilted T-snake defines a picture over Δ: at position $\mu(i)$ we find tile $\nu(i)$ hence the unit square at that position in the rectangle is labelled by $\varphi(\nu(i))$.

In this paper the mapping μ is never specified explicitly. Instead it is represented by a (meandering) walk over the units of a rectangle.

Picture languages are defined by additionally specifying initial and final tiles used in the snakes. Let $(\Sigma, T, I, F, \varphi)$ be such that (Σ, T, φ) is as before, and $I, F \subseteq T$ are sets of *initial* and *final* tiles, respectively. Then the perfectly quilted T-snakes of which the first tile $\nu(1)$ is chosen from I and the last tile $\nu(k)$ is chosen from F, define a picture language over Δ via the tile labelling φ. Such a language will be called a *perfectly quilted rectangular snake* language, or *PQRS* picture language for short.

Example 2. Consider the picture language of all odd sized squares with white and blue tiles such that the middle position is blue. A non-deterministic finite state automaton moving over a grid would accept these by first checking that the input is an odd sized square by moving along the diagonal from the SW-corner to the NE-corner. Then the middle position is (non-deterministically!) checked by moving from the NE-corner back to an arbitrary position along the diagonal, verifying this position is blue, then making a turn to the SE-corner. If this succeeds, the turning point was indeed the middle position. This idea is easily transferred to four-sided tiles, cf. the previous example.

Snakes cannot visit the same position twice. However, using perfect quilting it is possible to determine the center by spiraling inwards, cf. Fig. 3. In the case of an odd sized square this walk stops at the first step after a turn to the north. In the figure this corresponds to the third diagram.

We build the snake tiling system as follows, see Fig. 4. The edges are marked by $\{ \uparrow, \rightarrow, \downarrow, \leftarrow, \cdot, :, 1 \}$, where the symbols $\cdot$ and : are positioned in such a way that it is impossible to find a matching marking at that edge (since : occurs W and S, while $\cdot$ occurs E and N). Thus, along the snake only the arrows can be used to match the marking of adjacent tiles, and they are positioned as to indicate the direction in which the snake is continued. As always, $\sharp$ is the symbol marking the border of the rectangle, represented by a bold line in the diagrams. For each white tile one should also add the corresponding blue tile. The last tile however is only available in blue. It is designed to sit in the middle.

To complete the specification of the tiling system, we choose I to consist of the first tile given above (the SW-corner) in both colours, and we choose F to consist of the last tile (center) in blue only.

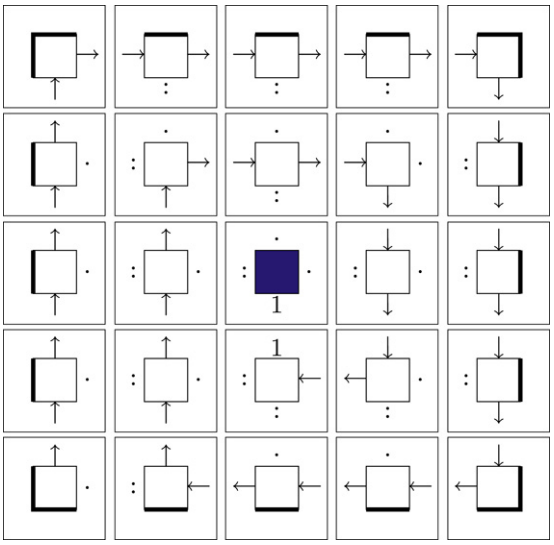


Fig. 5. A possible perfectly quilted snake tiling using the tiles from Fig. 4. The snake begins and ends with the specified tiles. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

As example we show in Fig. 5 an appropriate snake tiling for the 5×5 square with blue center unit. As discussed, tiles match along the snake (as required in the definition) but nowhere else. □

Normal form. In the basic definition of PQRS tilings above tiles are matched in such a way that their markings match on two sides along the snake, which is a single trail covering all tiles exactly once. This trail however is not explicitly coded in the tiles. A given tiling may admit several snakes if tiles happen to match on several sides, in particular, classical four-sided Wang tilings admit many snakes. It is easy to include the pattern that is traced by the snake in the tiling itself by considering so-called directed tiles, precisely as we have done in the example above. Here the matching edge markings indicate the direction the snake follows, whereas the edges of the tiles that are not on the trail of the snake do not match. This construction is detailed in [5]. Example 2 is in normal form, thanks to the design of non-matching symbols · and :.

Snake tilings viewed as regular descriptions. Thus in this normal form, the perfectly quilting snake traces a single continuous trail covering all the positions of the rectangle, with matching edges along the trail. This is very similar to the path of a finite state automaton walking over the rectangle and visiting each of the tiles exactly once. The tile set T describes the transitions of the automaton, the change of state (from incoming marking to outgoing marking), and a change of direction. The tiles from I and F correspond to initial and final transitions, while the symbol 'read' by the automaton is the 'colour' of the tile assigned by the labelling.

Like in the one-dimensional case, finite automata can be replaced by equivalent regular expressions. Here the expressions not only include the symbol (colour of the tile) but also the change of direction. Clearly not every string matching the regular description satisfies the 'PQRS' condition that every position should be visited exactly once. This is an additional, external requirement.

Regular expressions of moves over the plane are the main ingredient of chain code picture languages [13], where the expressions define simple drawings consisting of line segments. A *chain code picture language* is defined by a (regular) language over the alphabet $D = \{r, d, l, u\}$. These symbols are interpreted as instructions for a plotter which at each step draws a unit segment in the designated direction right, down, left, or up. For instance, d^2urud^2 results in the figure H, when we start in the NW-corner. Clearly snakes and chain code pictures are related, although there are several technical differences. For instance, a segment of a chain code picture may be traced twice (as in the H).

Many properties of chain code picture languages are undecidable. For instance, whether the language contains a picture without subfigure H [4], or whether it contains a self-avoiding picture [18] (interesting in the context of snakes!). Note that, emptiness of chain code picture languages is trivially decidable (every string over D defines a picture), while that for PQRS tilings is undecidable. This follows from Theorem 4 (and the fact that emptiness for recognizable picture languages is undecidable [8]).

3. Tilings for snakes

We show that every PQRS picture language is recognizable, but first we argue that this is not immediate. As before we assume the tiles are directed, each tile is superimposed with an arrow indicating the direction of the snake before and after the present tile. Now eliminate the markings from the other two edges, replacing them by a 'blank' marking (but we keep the

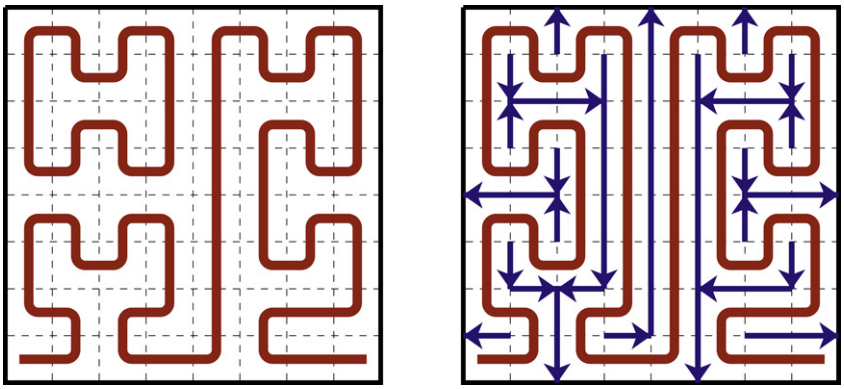


Fig. 6. Without precautions four-sided tilings do not automatically fix snakes (left) but when the corners of the unit squares are forming a forest then a snake is forced (right).

special marking for the border of the rectangle). In this way a snake tiling is automatically a four-sided tiling: the markings on all shared edges match.

Unfortunately not every tiling of a rectangle with matching arrows (and blank edges) defines a single perfectly quilted snake for the rectangle. First, there can be several separate snakes. This can be solved by counting the number of initial tiles, where snakes start. This number should be exactly one, and this can be determined by a finite state process coded in the edge markings. Second, tiles may form a single snake, together with several separate circular components, cf. Fig. 6. We have to avoid this, and force the arrows into a single path.

Recall that picture languages defined by tiling systems are characterized using existential monadic second-order logic [9]. However, the obvious way to define connectedness in monadic second-order logic uses universal quantification. (First define the property of a 'closed' set of tiles, i.e., that it contains for each of its tiles its successor along the snake. To define a path along the snake one additionally requires that the closed set is 'minimal', i.e., every closed set should contain the path.) As this is not allowed in the existential part of the logic, at a first glance this makes our task impossible.

Surprisingly, Reinhardt [16] shows that the language of $\{a, b\}$-labelled pictures in which the b's form a connected area is recognizable, which means that connectedness *can* be defined for pictures without universal quantification. The trick is to 'grow' two sets of trees. One set covers the area of b's, the other set of 'tentacles' connects (the bottom-right corner of) each square to one of the outer walls. Separately, neither the b-tree nor the tentacle trees can be guaranteed to be without cycles, but together the one forces the other to be a tree. This method can also be used to define a connected snake, illustrated in Fig. 6, to the right.

Theorem 3. *Every PQRS picture language is recognizable.*

Proof. The proof follows that of Reinhardt [16]. We assume we have a tiling system (with initial and final tiles) the perfectly quilted tilings of which define a picture language. We assume the tilings system is in the normal form we discussed: the edge markings are such that that the tiles only match on two sides, on the edges that follow the snake.

We build a new tiling system for the same language by adding additional decoration to the edges of the tiles. This decoration ensures that a proper (four-sided) tiling of the rectangle defines a single snake of the original tiles, without circular components.

The additional decoration defines an orientation for each side of the tile which is not labelled by a direction for the snake. This orientation should match the one of the neighbouring tiles, and together these edges of tiles should form a forest of 'tentacles' (terminology of Reinhardt) connecting each corner of the tiles to the border of the rectangle. Note that the snake arrows are perpendicular to the sides of the tile (snakes cross the sides) whereas the orientation for the tentacles is parallel to the sides (tentacles run in between the tiles).

In general a tree structure requires that each node has a unique 'parent'. Here we ensure this by orienting the corners (in one of the directions up, right, down, left) indicating the unique direction of an arrow leaving that corner; all other adjacent sides (when not used by the snake) are oriented such that they have incoming arrows.

As an example, for a tile where the snake enters from the left and leaves upwards, there are twelve possible orientations of the corners and matching orientations of the two remaining sides. These are depicted in Fig. 7.

The borders of the rectangle do not have orientations. They are labelled with ♯ as customary. We do require that the edges perpendicular to the borders have arrows leading to the border. In this way the border defines the root for a particular tentacle tree. Fig. 8 shows the tiles with corner orientations for part of the rectangle from Fig. 6.

Now ideally, we have a single snake and a forest of edge arrows connecting each corner to the border of the rectangle. In case the path of the snake defines a single linear component and one or more cycles, there are corners that are not connected to the border. As all corners point in some direction, and as edge orientations do not cross the snake arrows, there must be a cycle following the edges of the tiles within the cycle of the snake. Now we iterate the argument. Within the cycle of the arrows along the edges must be another snake cycle, etc. Eventually we reach the situation where there is not sufficient

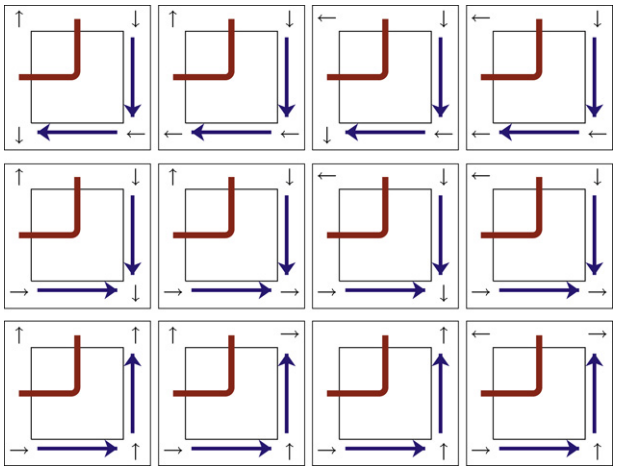


Fig. 7. Starting with a single tile we obtain twelve possible tiles with different orientations of the edges where the snake does not pass.

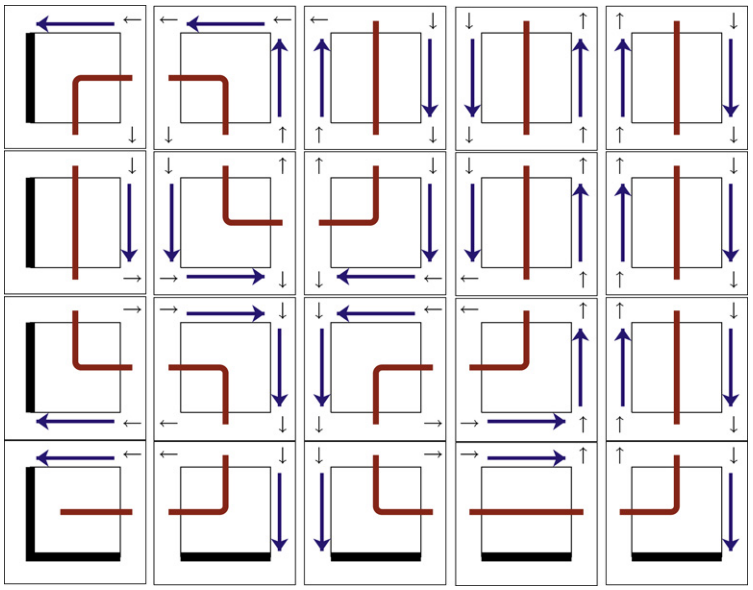


Fig. 8. Four-sided representation of the SW-corner of the snake from Fig. 6 (right), using tiles as specified in the proof of Theorem 3.

space to accommodate a cycle. This contradicts the tiling pattern: every arrow must point to a successive arrow. Hence, there cannot be cycles (in either snake or 'tentacles' along the edges). □

4. Weaving snakes

Tilings defined by snakes are only required to fit along the snake: two edges for each tile. The connectivity of the classic Wang tiling systems is greater as they have to fit in four directions. Still we show that every four-sided tiling can be defined using PQRS tilings provided we cheat a little, using tiles larger than unit size. Super-tiles (as explained in [12]) make it possible to code edge colours by 'bumps and dents'. The snake follows a basic meander, where consecutive rows match according to form (rather than edge marking).

The k-multiplication of the $m \times n$ picture p is the $km \times kn$ picture p' such that the label $p'(i, j)$ equals $p(i/k, j/k)$, where / denotes integer division (see Fig. 9 for a 2-multiplication). The operation is extended to languages.

Theorem 4. *For every recognizable picture language L there exists a constant k such that its k-multiplication is a PQRS picture language.*

Proof. Consider now the k-multiplication of a picture language L defined by a tiling system (for a suitable value k to be determined).

The main route of the snake we will build is a meander following the super-tiles, (almost) filling each of them in succession. This allows to check that the super-tiles fit in one direction (the main direction taken by the snake). The markings

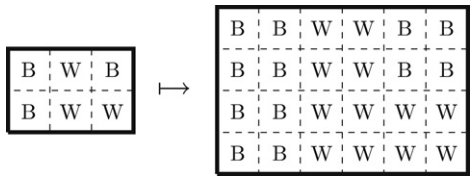


Fig. 9. A 2-multiplication.

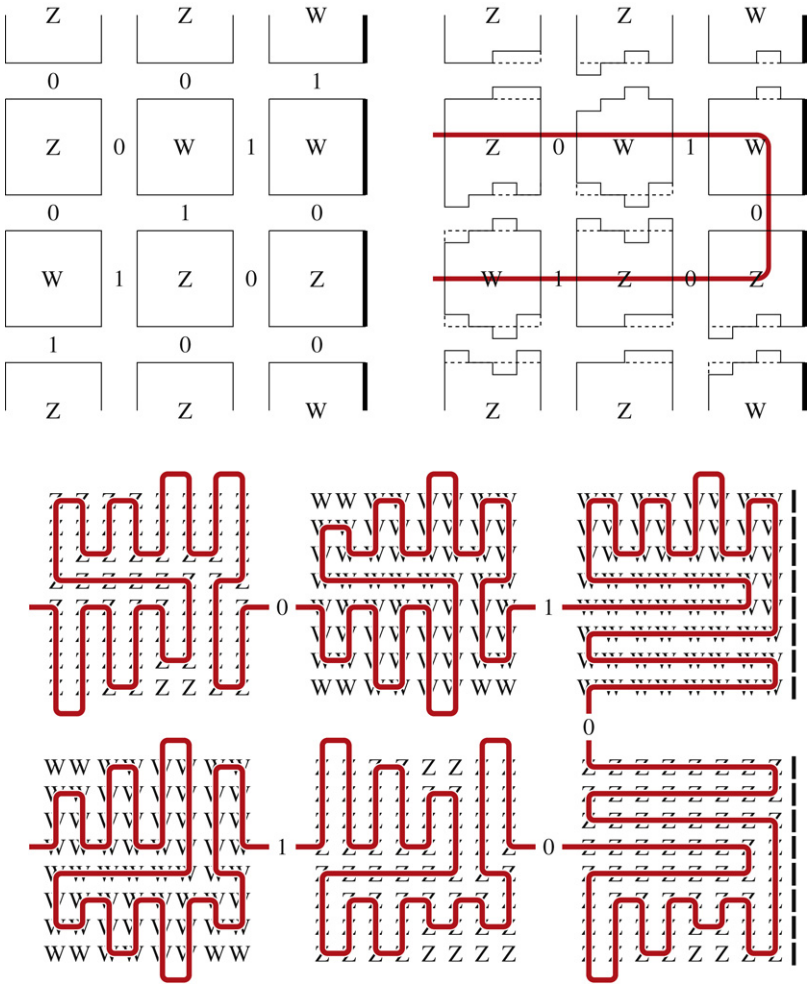


Fig. 10. Six adjacent tiles with matching edges markings are converted into snakes. In the second diagram the top and bottom edge markings have been replaced by edge shapes. Matching shapes represent to matching edges (and a pair of tile colours). In the third diagram the tiles have been enlarged to 8×8 super-tiles formed by snakes that form the edge shapes.

of the edges in the other direction is left as a trail of bumbs and dents. For each $k \times k$ super-tile we have to design (around) k^2 separate unit size tiles such that they can only fit to form the path of the snake over the super-tile (plus/minus some tiles along the edges). In order to force the small tiles that make up a super-tile into a fixed trail all edge markings for the small tiles are new, except the first and last marking along the trail which are copied from the original tile. The colours assigned to these small tiles are all the same (except when the snake moves to the neighbouring super-tile).

In Fig. 10 we show how six adjacent tiles are converted into six 8×8 super-tiles. The main direction of the snake is horizontal, with turns at the sides.

All positions in a super-tile have the same colour. While meandering along the edge of the super-tile, the (small) tiles have to match the current label, but sometimes that of the neighbouring super-tile, when the edge is crossed. This means the form of the trail also codes for the label of the neighbouring tile. We need a little math here. In a $k \times k$ super-tile (with $k \geq 4$ and k even) along a single border we can code $k/2$ bumps, giving (at least) $2^{k/2}$ possibilities for the form of the trail along the border. With c edge markings and t tile labels of the original tiling system, the trail should distinguish $c \cdot t^2$

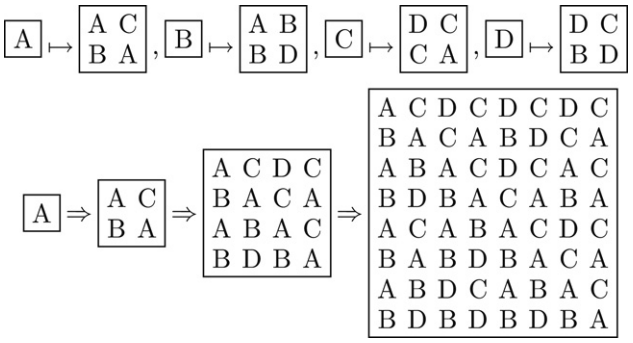


Fig. 11. The chair substitution and the first four of the resulting successive tilings.

possibilities, which can be obtained by choosing an appropriate k. Note again that around k^2 unique tiles with fresh markings are necessary to program the trail in such a super-tile. □

Substitution tilings. Another popular way to define sets of pictures are *substitution tilings*. In general these can be used to obtain tilings of the plane by geometric forms, like the Penrose tilings. Here we consider the two-dimensional version of string morphisms, and replace each square in a rectangle by an $k \times \ell$ rectangle which is determined by the label of the original square. In this way a rectangle is changed into a larger rectangle. This process can be iterated, starting from a simple rectangle, to obtain a picture language. In this way one may generate pleasing fractal pictures, see, e.g., Chapter 5.4 in [20] or Chapter 5 in [6].

Example 5. An example is given in Fig. 11 where each symbol is replaced by a particular 2×2 square, a substitution called the *square chair*, presented in [2], see [7] for a visualization. □

It follows from the work of Mozes [14] that each (two-dimensional) substitution tiling has an equivalent four-sided tiling system with 'matching rules', like the one explicitly constructed by Kari [11], see also the tiling that underlies the constructions of Reinhardt [17]. We stress here that we only consider tilings of (finite) rectangles, rather than of the complete plane. This simplifies constructions as it is easy to force an initial tile to start a proper pattern.

Proposition 6 (*[14]*). *Every picture language defined by a substitution tiling is recognizable.*

For this specific type of recognizable tilings we can strengthen the correspondence between recognizable picture languages and PQRS picture languages: here no multiplication is needed. We simplify the exposition by considering 2×2 substitutions only. Although the result can be generalized, it cannot be shown using our methods for all sizes of the substituted rectangles. In particular, our snakes for super-tiles do not fit odd length squares.

Theorem 7. *Every picture language defined by a 2×2 substitution tiling is a PQRS picture language.*

Proof. Starting with a substitution h into 2×2 squares, consider the tiling system T defined by h, cf. Proposition 6. For a suitable multiplication k, the k-multiplication of T is a PQRS picture language, cf. Theorem 4. We may take multiplication factor k to be of the form 2^ℓ.

For a symbol σ let $h^\ell(\sigma)$ be the $2^\ell \times 2^\ell$ tiling which results from ℓ times iterating substitution h starting with a single square labelled by σ.

In the tiling system for the 2^ℓ-multiplication each $2^\ell \times 2^\ell$ super-tile consists of unit squares labelled by the same symbol. Now, we replace each $2^\ell \times 2^\ell$ super-tile labelled by symbol σ by the $2^\ell \times 2^\ell$ tiling $h^\ell(\sigma)$. Thus the super-tile is labelled by an iteration of the substitution h. In this way we obtain a tiling which again is defined by the substitution h: the multiplication is replaced by an iteration of the substitution!

Returning to the snake construction for tiling systems, instead of checking a single label σ in every position of the super-tile the snake may check the corresponding position of a super-tile $h^\ell(\sigma)$ (and we can build that image into the 'finite state' of the snake tiles).

Only the initial steps of the substitution process, tiles smaller that the multiplication factor, are not covered by this construction. For this finite number of rectangles separate snake tiles have to be designed, and added to the set already obtained. □

5. Back to Hilbert

Also the Hilbert curve we started with can be obtained from a single tile by iterating a two-dimensional substitution. One needs twelve tiles and rules as in Fig. 12, a mapping already implicit in the definition of Hilbert [10].

So we have the rather amusing conclusion that the snake-like Hilbert pictures can be defined using a PQRS tiling. This in a sense is a regular one-dimensional description (with additional perfect quilting requirement). However, the PQRS snake tiles do not follow the natural 'flow' of the Hilbert curve (the snake follows a zig-zag along consecutive super-tiles). The

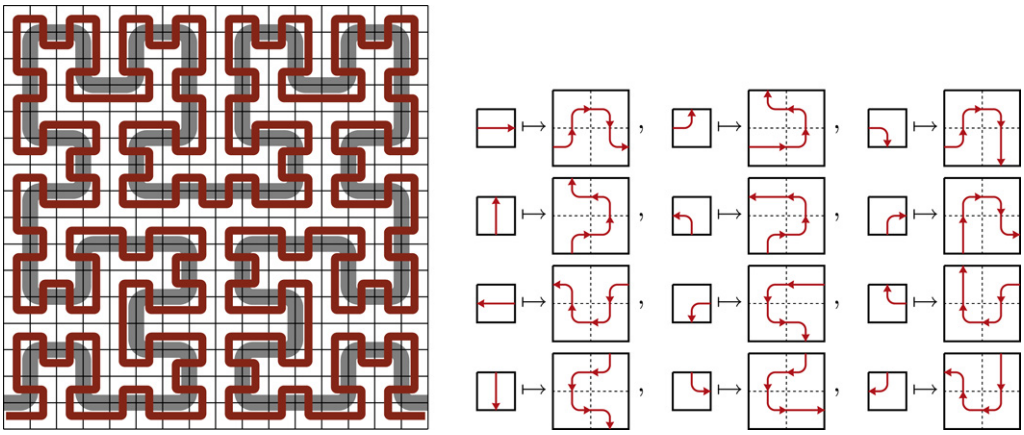


Fig. 12. Two levels of the Hilbert tilings, and a set of substitution rules for the Hilbert tilings.

question that remains is whether the Hilbert curves can be defined as snakes (with perfect quilting), so that the curve 'follows' its description. This even might have some implications for the design of DNA tiles that could construct a physical implementation of the curve [19].

Acknowledgements

We are indebted to Klaus Reinhardt and Jarkko Kari for providing helpful information. We thank the referees for suggestions that improved our presentation.

References

[1] L.M. Adleman, J. Kari, L. Kari, D. Reishus, On the decidability of self-Assembly of infinite ribbons, in: Proceedings 43rd Symposium on Foundations of Computer Science, FOCS 2002, 2002, pp. 530–537. doi:10.1109/SFCS.2002.1181977.
[2] M. Baake, R.V. Moody, M. Schlottmann, Limit-(quasi-)periodic point sets as quasicrystals with p-adic internal spaces, Journal of Physics A: Mathematical and General 31 (1998) 5755–5765. doi:10.1088/0305-4470/31/27/006.
[3] M. Blum, C. Hewitt, Automata on a 2-dimensional tape, in: Proceedings 8th Symposium on Switching and Automata Theory, SWAT, 1967, pp. 155–160.
[4] J. Dassow, F. Hinz, Decision problems and regular chain code picture languages, Discrete Applied Mathematics 45 (1993) 29–49. doi:10.1016/0166-218X(93)90138-E.
[5] Y. Etzion-Petruschka, D. Harel, D. Myers, On the solvability of Domino snake problems, Theoretic Computer Science 131 (1994) 243–269. doi:10.1016/0304-3975(94)90174-0.
[6] F. Drewes, Grammatical Picture Generation, A Tree-Based Approach, Springer, 2006.
[7] D. Frettlöh, E. Harriss, Tilings Encyclopedia. Available online at: tilings.math.uni-bielefeld.de.
[8] D. Giammarresi, A. Restivo, Two-dimensional Languages, in: Handbook of Formal Languages, vol. 3, 1997, pp. 215–267.
[9] D. Giammarresi, A. Restivo, S. Seibert, W. Thomas, Monadic second-order logic over rectangular pictures and recognizability by tiling systems, Information and Computation 125 (1996) 32–45. doi:10.1006/inco.1996.0018.
[10] D. Hilbert, Ueber die stetige Abbildung einer Linie auf ein Flächenstück, Mathematische Annalen 38 (1891) 459–460. doi:10.1007/BF01199431.
[11] J. Kari, Reversibility and surjectivity problems of cellular automata, Journal of Computer and Systems Sciences 48 (1994) 149–182. doi:10.1016/S0022-0000(05)80025-X.
[12] J. Kari, Infinite snake tiling problems, in: Proceedings Developments in Language Theory 2002, in: Lecture Notes in Computer Science, vol. 2450, 2003, pp. 67–77. doi:10.1007/3-540-45005-X_6.
[13] H.A. Maurer, G. Rozenberg, E. Welzl, Using string languages to describe picture languages, Information and Control 54 (1982) 155–185. doi:10.1016/S0019-9958(82)80020-X.
[14] S. Mozes, Tilings, substitution systems and dynamical systems generated by them, Journal d'Analyse Mathématique 53 (1989) 139–186. doi:10.1007/BF02793412.
[15] L. de Prophetis, S. Varricchio, Recognizability of rectangular pictures by wang systems, Journal of Automata, Languages and Combinatorics 2 (1997) 269–288.
[16] K. Reinhardt, On some recognizable picture-languages, in: L. Brim, J. Gruska, J. Zlatuska (Eds.), Proceedings 23rd International Symposium Mathematical Foundations of Computer Science, MFCS 1998, in: Lecture Notes in Computer Science, vol. 1450, 1998, pp. 760–770. doi:10.1007/BFb0055753.
[17] K. Reinhardt, The #a = #b pictures are recognizable, in: A. Ferreira, H. Reichel (Eds.), Proceedings 18th Annual Symposium on Theoretical Aspects of Computer Science, STACS 2001, in: Lecture Notes in Computer Science, vol. 2010, 2001, pp. 527–538. doi:10.1007/3-540-44693-1_46.
[18] D. Robilliard, D. Simplot, Undecidability of existential properties in picture languages, Theoretical Computer Science 233 (2000) 51–74. doi:10.1016/S0304-3975(97)00279-X.
[19] P.W.K. Rothemund, N. Papadakis, E. Winfree, Algorithmic self-assembly of DNA Sierpinski triangles, PLoS Biology 2 (12) (2004) e424. doi:10.1371/journal.pbio.0020424.
[20] S. Wolfram, A New Kind of Science, Wolfram Media, 2002. Also online via: http://www.wolframscience.com.