Labelled Polyomino Codes: A Survey of Algebraic and Decidability Issues

Włodzimierz Moczurad

Nowy Sacz School of Business — National-Louis University
Zielona 27, PL-33-300 Nowy Sacz
moczurad@wsb-nlu.edu.pl

Abstract. Variable-length word codes, i.e., sets of words such that every word generated has a unique factorization over the set, are a common object of study. Tilings, including polyomino tilings, are another research classic. Here we consider two-dimensional structures, labelled polyominoes ("bricks"), that can be viewed as a natural extension of both words (or word codes) and polyominoes. We begin with basic definitions and properties related to codicity, including a two-dimensional version of the well-known Schützenberger's theorem. We then continue with decidability results, the main one being the undecidability of codicity testing. We prove that with the polyominoes restricted to squares, the codicity of small sets (two bricks) is decidable, but 15 bricks are enough to make the problem undecidable. Thus the frontier between decidability and undecidability lies between these two numbers.

1 Introduction

In the paper we consider labelled polyominoes (also called bricks). They are two-dimensional structures that can be viewed as a natural extension of both words (or word codes) and polyominoes. An in-depth systematic introduction to the theory of codes can be found in [3], whilst [1, 4] make a good introductory reading on polyomino tilings. Beginning with basic definitions and properties related to codicity, including a two-dimensional version of the well-known Schützenberger's theorem, we continue with decidability results, the main one being the undecidability of codicity testing. We prove that with the polyominoes restricted to squares, the codicity of small sets (two bricks) is decidable, but 15 bricks are enough to make the problem undecidable.

Let A be a finite alphabet of labels. A *labelled polyomino* (or a *brick*) is a partial mapping $k : \mathbb{Z}^2 \to A$, where $\operatorname{dom} k$ is finite and connected. $D \subseteq \mathbb{Z}^2$ is connected if for every pair of points $(x, y), (x', y') \in D$ there exists a path (x_0, y_0), (x_1, y_1), ..., $(x_n, y_n) \in D$ such that $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (x', y')$ and $|x_{i+1} - x_i| + |y_{i+1} - y_i| = 1$ for $i = 0, 1, ..., n - 1$.

Note that a brick can be viewed as a polyomino with its cells labelled with the symbols of A. If $|A| = 1$, there is an obvious natural correspondence between bricks and polyominoes. The set of all bricks over A is denoted by $A^{\bowtie}$.

Given a set of bricks $X \subseteq A^{\bowtie}$, the set of all bricks tilable with (translated copies of) the elements of X is denoted by $X^{\bowtie}$. Note that we do not allow rotations of bricks. $X \subseteq A^{\bowtie}$ is a *brick code,* if every element of $X^{\bowtie}$ admits exactly one tiling with the elements of X.

More formally, given a brick k and a vector $\boldsymbol{u} = (x_u, y_u) \in \mathbb{Z}^2$, the *translation* of k is the brick $k' = \boldsymbol{u}k$ such that $\operatorname{dom} k' = (x_u, y_u) + \operatorname{dom} k$ and $k'(x, y) = k(x - x_u, y - y_u)$. Now for $k \in X^{\bowtie}$, a *factorization* of k over X is a family $\{(\boldsymbol{u_i}, k_i)\}_{i=1,\ldots,n}$ such that $k = \bigcup_{i=1}^{i=n} \boldsymbol{u_i}k_i$, with all $\operatorname{dom} \boldsymbol{u_i}k_i$ pairwise disjoint. X is a *brick code,* if for each $k \in X^{\bowtie}$ there is exactly one factorization of k over X. In this context, a factorization is also called a *tiling.*

The *effective alphabet* of $X \subseteq A^{\bowtie}$ is the set of all symbols that appear on bricks in X, i.e., $\bigcup_{k \in X} k(\operatorname{dom} k)$. If $k \in A^{\bowtie}$ is a square brick, then by $\operatorname{len} k$ we denote the edge length of k, i.e., $\sqrt{|\operatorname{dom} k|}$.

Given a rectangular brick $t \in A^{\bowtie}$, by $t^{\,p \times q}$ we denote a brick obtained by stacking together p copies of t vertically, and then q copies of this compound horizontally.

We use the following pictorial representation of bricks. Each point of the domain, $(x, y) \in \operatorname{dom} k$, is drawn as a square $[x, x + 1] \times [y, y + 1] \subseteq \mathbb{R}^2$. The label $k((x, y))$ appears inside the square. Since we usually consider bricks up to translation, we do not mark the absolute coordinates.

Example 1. Let $k_1, k_2, k_3 \in \{a, b\}^{\bowtie}$. Now define:

- $\operatorname{dom} k_1 = \{(0, 0)\}$ and $k_1((0, 0)) = a$
- $\operatorname{dom} k_2 = \{(0, 0), (1, 0), (2, 0), (0, 1), (2, 1)\}$ and $k_2((0, 1)) = k_2((2, 0)) = a$ and $k_2((0, 0)) = k_2((1, 0)) = k_2((2, 1)) = b$
- $\operatorname{dom} k_3 = \{(0, 0), (1, 0), (1, -1), (1, 1)\}$ and $k_3((1, -1)) = k_3((1, 1)) = a$ and $k_3((0, 0)) = k_3((1, 0)) = b$.

The representation of k_1, k_2 and k_3 is shown below.

k_1: $\boxed{a}$ $\qquad$ k_2: $\begin{array}{|c|c|c|}\hline a & & b \\ \hline b & b & a \\ \hline \end{array}$ $\qquad$ k_3: $\begin{array}{|c|c|}\hline & a \\ \hline b & b \\ \hline & a \\ \hline \end{array}$

2 Brick codes

Proposition 1. *$X \subseteq A^{\bowtie}$ is a brick code iff every finite subset of X is a brick code.*

Proof. The "only if" part is obvious. If X is not a brick code, then there exists a brick with two different factorizations, each of them containing a finite number of bricks. Taking these bricks, we obtain a finite subset of X which is not a code. □

Proposition 2. *Let $|A| = n \geq 2$ and let $B = \{a, b\}$. There exists a mapping $f : A^{\bowtie} \to B^{\bowtie}$ such that for any set of bricks $X \subseteq A^{\bowtie}$ we have: X is a brick code (over A) iff $f(X)$ is a brick code (over B).*

Proof. Take $m = \lceil \log_2 n \rceil$ and note that there are at least n different $1 \times m$ rectangular bricks over B; these are essentially binary numbers $0...n-1$. The numbers can be used to identify the letters of A. Now f is constructed in such a way that each point of $A^{\bowtie}$ is mapped to a $1 \times m$ rectangle labelled with an appropriate number. It is easily seen that f preserves the codicity. □

The above property allows simulating an arbitrarily large alphabet with a two-letter set. Actually, shapes alone (i.e., a single-letter alphabet) are sufficient for the simulation.

Proposition 3. *For any mapping $\pi : A \to B$ and a set $X \subseteq A^{\bowtie}$, if $\pi(X)$ is a code, X is also a code.*

Proof. Assume that $X \in A^{\bowtie}$ is not a code. Then there exists $k \in X^{\bowtie}$ with two different factorizations: $k = \bigcup_{i=1}^{i=n} \boldsymbol{u_i} x_i = \bigcup_{j=1}^{j=m} \boldsymbol{v_j} y_j$. Consider $k' = \pi(k) \in B^{\bowtie}$. Obviously, $k' = \bigcup_{i=1}^{i=n} \boldsymbol{u_i} \pi(x_i) = \bigcup_{j=1}^{j=m} \boldsymbol{v_j} \pi(y_j)$. Now we claim that these two factorizations of k' are different. If they are not, then $n = m$ and $\boldsymbol{u_i} = \boldsymbol{v_i}$, $\pi(x_i) = \pi(y_i)$ for $i = 1, ..., n$. Consequently, the only way in which the original factorizations of k can be different is to have different labellings. But the labelling is determined by k itself. Thus we have two different factorizations of k', and $\pi(X)$ is not a code. □

This property can be interpreted as follows: if $\pi(X)$ contains enough information to determine the factorization of every brick from $\pi(X)^{\bowtie}$, then there is certainly enough information in X itself to find the factorization of every brick from $X^{\bowtie}$. The converse does not usually hold, of course.

Let $\pi : A \to \{b\}$ and let $X \subseteq A^{\bowtie}$ be a code. We say that X is a *shape code,* if $|\pi(X)| = |X|$ and $\pi(X)$ is a code. In other words, X is a shape code if it remains a code regardless of the labelling of the bricks.

Example 2. The set X depicted below is a code over $A = \{a, b\}$. Taking $\pi : A \to \{b\}$, we obtain $\pi(X)$ which is not a code.

2.1 Uniform sets

A set $X \subseteq A^{\bowtie}$ is called *uniform,* if all its elements have identical shape, i.e., $\forall k, l \in X : \operatorname{dom} k = \operatorname{dom} l$. X is additionally called *full,* if it contains all possible bricks of a given shape. A uniform set is obviously a code. The following properties reflect the idea of bricks as shapes with added labels.

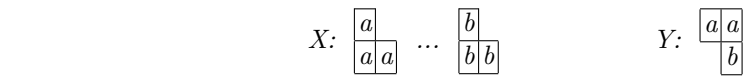
Proposition 4. *Assume that $X, Y \subseteq A^{\bowtie}$ are uniform and let $k_0 \in X$, $l_0 \in Y$. If $\{k_0, l_0\}$ is a shape code then $\{k, l\}$ is a shape code for any $k \in X, l \in Y$, and $X \cup Y$ is a code.*

Proof. Let π be a mapping $\pi : A \to \{b\}$ and let $m \in (X \cup Y)^{\bowtie}$. Denote $m' = \pi(m)$, $k' = \pi(k_0)$ and $l' = \pi(l_0)$. Then $k' = \pi(k)$ for any $k \in X$ and $l' = \pi(l)$ for any $l \in Y$ since X, Y are uniform. Consequently, $m' \in \{k', l'\}^{\bowtie}$. Since $\{k', l'\}$ is a code, there is a unique factorization of m' into k' and l'. Both sets X, Y are uniform, so now we can reconstruct the labelling of m. Thus $X \cup Y$ is a code. □

Proposition 5. *Assume that $X, Y \subseteq A^{\bowtie}$ are uniform and full, $X \neq Y$. If $X \cup Y$ is a code, then $\{k, l\}$ is a shape code for any $k \in X$, $l \in Y$.*

Proof. Let π be a mapping as above. Assume that $\{k, l\}$ is not a shape code for some $k \in X$, $l \in Y$. Then there exists $m' \in \{k', l'\}^{\bowtie}$ such that $m' = \bigcup \boldsymbol{u_i} x'_i = \bigcup \boldsymbol{v_i} y'_i$ with $x'_i, y'_i \in \{k', l'\}$. Now, label m' arbitrarily. As X, Y are full, we can reconstruct labels for $x_i, y_i \in X \cup Y$. We have $m = \bigcup \boldsymbol{u_i} x_i = \bigcup \boldsymbol{v_i} y_i$ so $X \cup Y$ is not a code. □

Example 3. Consider a full uniform set $X \subseteq \{a, b\}^{\bowtie}$ containing bricks with $\operatorname{dom} k = \{(0,0), (1,0), (0,1)\}$ and Y containing a single element l with $\operatorname{dom} l = \{(0,1), (1,0), (1,1)\}$. $X \cup Y$ is a code, but $\{k, l\}$ (for any $k \in X$) is not a shape code.



2.2 Two-element sets of dominoes

Testing whether a given set $X \in A^{\bowtie}$ is a brick code is undecidable in the general case (see next Section). Below we present a simple case of a two-element set of domino-shaped bricks.

Theorem 1. *Let X contain two domino-shaped bricks, k_1 with $\operatorname{dom} k_1 = \{(0,0), (0,1)\}$ (vertical domino) and k_2 with $\operatorname{dom} k_2 = \{0, 1, ..., p-1\} \times \{(0,0)\}$ (horizontal domino of length $p \geq 1$). Then X is a code iff $|\operatorname{alph}(X)| > 1$.*

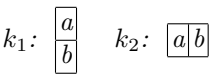
Proof. Certainly, if $|\operatorname{alph}(X)| = 1$ then X is not a code (in other words, X is not a shape code). A double factorization can be constructed e.g. for a $2 \times p$ rectangle by taking either p vertical dominoes or two horizontal ones (cf. [1]).

To prove the converse it suffices to show that X is a code for any labelling that gives $|\operatorname{alph}(X)| = 2$, since being a code with $|\operatorname{alph}(X)| = 2$ implies being a code for any labelling with $|\operatorname{alph}(X)| > 2$. Indeed, suppose that this has been proven and take $\operatorname{alph}(X) = \{a_1, a_2, ..., a_n\}$ ($n \geq 2$). Consider the projection $\pi : \{a_1, a_2, ..., a_n\} \to \{a, b\}$ with $\pi(a_1) = a$ and $\pi(a_i) = b$ for $i = 2, ..., n$. As $\pi(X)$ is a code, X is also a code.

Now assume that $\operatorname{alph}(X) = \{a, b\}$ and consider the following cases, together with all symmetries:

- $\operatorname{alph}(k_1) = a$, and $\operatorname{alph}(k_2) = b$ or $\operatorname{alph}(k_2) = \{a, b\}$: The position of k_2 can always be determined by finding the label b, thus X is a code. Note that the assumption on the shape of the bricks is not important in this case.

– $\mathrm{alph}(k_1) = \{a, b\}$ and $\mathrm{alph}(k_2) = \{a, b\}$: Let $p = 2$, i.e., the horizontal domino k_2 has length 2; bricks with $p > 2$ can be dealt with in a similar way. Without loss of generality we can assume that X is as follows:



Assume that X is not a code, so there exists a brick with two different decompositions. We can take minimal decompositions, in the sense that they do not contain superfluous elements $(\boldsymbol{u_i}, x_i)$, $(\boldsymbol{v_j}, y_j)$ such that $\boldsymbol{u_i}x_i = \boldsymbol{v_j}y_j$. Call the decompositions D_1 and D_2. One of them, e.g., D_1, must contain k_1. Then D_2 has to have a brick or bricks which "cover it up." None of them can be k_1, since k_1 cannot overlap itself; thus k_1 in D_1 is covered by k_2's in D_2. But this requires two k_2 bricks placed as shown below.

Now we see that D_1 has to contain k_1 once again, extending the staircase-like shape e.g. to the left. Continuing in this way, we obtain an infinite sequence of steps, which contradicts the existence of a brick with two decompositions. □

2.3 Maximal and complete brick codes

A code $X \subseteq A^{\bowtie}$ is *maximal* over A, if it is not properly contained in any other code from $A^{\bowtie}$, i.e., for any code $Y \subseteq A^{\bowtie}$, $X \subseteq Y$ implies $X = Y$. Any code $X \subseteq A^{\bowtie}$ is contained in some maximal code from $A^{\bowtie}$. The proof of the analogous property for word codes (which uses the Zorn's Lemma to show that any set of codes contains a maximal element; see [3]) can be applied to brick codes as well.

For any set $X \subseteq A^{\bowtie}$ we define the *set of factors* of X as $F(X) = \{k \in A^{\bowtie} \mid \exists l_1, l_2, ..., l_n \in A^{\bowtie} : k \cup \bigcup_{i=1}^{n} l_i \in X\}$. In other words, k is a factor of X if it can be embedded in some brick from the set X. A set $X \subseteq A^{\bowtie}$ is *complete,* if $F(X^{\bowtie}) = A^{\bowtie}$.

A basic theorem of the theory of codes (a part of the Schützenberger's theorem, [16]) says that any maximal code is complete. We will prove a brick equivalent.

Two bricks $k, l \in A^{\bowtie}$ are *compatible,* if they are equal on the intersection of their domains, i.e., $k|\mathrm{dom}\, k \cap \mathrm{dom}\, l = l|\mathrm{dom}\, k \cap \mathrm{dom}\, l$.

A brick $k \in A^{\bowtie}$ is *self-overlapping,* if there exists a vector $\boldsymbol{u} \neq (0, 0)$ such that k and $\boldsymbol{u}k$ are compatible and $\mathrm{dom}\, k \cap \mathrm{dom}\, \boldsymbol{u}k \neq \emptyset$. A brick which is not self-overlapping is called non-overlapping.

Lemma 1. *Assume that $|A| > 1$. For any $k \in A^{\bowtie}$, there exists $l \in A^{\bowtie}$ such that $k \cup l$ is non-overlapping.*

Proof. Assume that $a, b \in A$. Note that the following brick, named D_4, is non-overlapping:

a	b	b	a
b	a	a	a
a	b	b	b
b	b	b	b

More formally,

- $\operatorname{dom} D_4 = I_4 \times I_4$
- points labelled with a: $D_4^{-1}(\{a\}) = \{(0,1),(1,2),(2,2),(3,2),(0,3),(3,3)\}$
- points labelled with b: $D_4^{-1}(\{b\}) = (\operatorname{dom} D_4) \setminus D_4^{-1}(\{a\})$.

Furthermore, an arbitrarily large non-overlapping brick can be constructed with the aid of the following induction: use D_4 as a 4×4 pattern and stack together 16 bricks by putting D_4 where the pattern has the label a, and B_4 where the label is b (B_m denotes an $m \times m$ brick labelled uniformly with b's). We obtain D_{4^2} in this way. By induction, $D_{4^{n+1}}$ is constructed in the same way, by stacking D_{4^n} and B_{4^n}. It is easy to verify that the bricks D_{4^n} $(n \geq 1)$ are non-overlapping.

Assume by contradiction that D_{4^n} is self-overlapping for some $n \geq 1$. Then there exists a vector $\boldsymbol{u} \neq (0,0)$ such that D_{4^n} and $\boldsymbol{u}D_{4^n}$ are compatible and $\operatorname{dom} D_{4^n} \cap \operatorname{dom} \boldsymbol{u}D_{4^n} \neq \emptyset$. Let $m = \max\{n \mid \exists p, q \in \mathbb{Z} : \boldsymbol{u} = (p4^n, q4^n)\}$; notice that $m < n$. If $m = n - 1$, we have a "top-level pattern collision:" one of the $D_{4^{n-1}}$ fields coincides with a $B_{4^{n-1}}$ field. But this implies that the pattern D_4 is self-overlapping, which is false. If $m < n - 1$, we have a similar collision at a lower level.

Now take any $k \in A^{\bowtie}$. If k is non-overlapping, we have nothing to do. Embed k in the smallest possible $4^n \times 4^n$ square by surrounding it with b's. If the square is now non-overlapping, we are done. Otherwise, take D_{4^n} and put it e.g. atop the square. Again it can be easily verified that this procedure leads to a non-overlapping brick containing the factor k. □

Theorem 2. *Assume $|A| > 1$. Any maximal brick code over A is complete.*

Proof. Assume that $X \subseteq A^{\bowtie}$ is not complete, i.e., $F(X^{\bowtie}) \neq A^{\bowtie}$. Take $k \in A^{\bowtie} \setminus F(X^{\bowtie})$. If k is self-overlapping, use the above Lemma to replace it with $k' = k \cup l$ which is non-overlapping; certainly, k' is not a factor of $X^{\bowtie}$ then.

Now we claim that $X \cup \{k'\}$ is a code. If not, there exists a (minimal) brick in $(X \cup \{k'\})^{\bowtie}$ with two different factorizations, both of them containing k'. Since k' is non-overlapping, this implies that k' is composed of factors of $X^{\bowtie}$. Thus k' is itself a factor of $X^{\bowtie}$. Hence $X \cup \{k'\}$ is a code and X is not maximal. □

2.4 Infix sets and codes

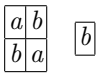
There is no natural brick extension of the symmetrical notions of prefixity and suffixity. However, we can define the notion of infixity which is an extension of word infixity as given in e.g. [5, 17].

We say that a set $X \subseteq A^{\bowtie}$ is *infix,* if $\forall k, l \in X, k \neq l \; \neg\exists \boldsymbol{u} : (\boldsymbol{u} + \operatorname{dom} k) \subseteq \operatorname{dom} l$ and $\boldsymbol{u}k, l$ are compatible. That is, no brick is a "subbrick" of another,

or—in terms of factors—$\forall k \in X : F(\{k\}) \cap X = \{k\}$. A set $X \subseteq A^{\bowtie}$ is *strictly infix,* if $\forall k, l \in X, k \neq l \ \neg\exists \boldsymbol{u} : \boldsymbol{u}k, l$ are compatible. This means that no two bricks in X overlap. Note that an infix set is not necessarily a code. However, a set that is strictly infix is clearly a code.

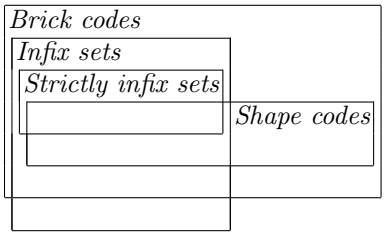
Example 4. The following set is infix but is not a code.

Example 5. The following set is a code which is neither infix, nor a shape code.



We now have a classification of brick sets which is depicted in the diagram below. Note that all inclusions are strict.

Proposition 6.



3 Brick code decidability

The problem of whether a given set of bricks (even polyominoes) is a code is undecidable in general. We prove this by reduction from the Wang tilabilty problem (cf. [2, 14]). The problem is open for two-element sets. We then consider sets consisting of square bricks only. We show that in this setting, the codicity of small sets (two bricks) is decidable, but 15 bricks are enough to make the problem undecidable.

Note that the codicity problem is trivially decidable if the squares use just one label, i.e., when they are effectively polyominoes. Only singleton sets are codes then. Thus in the sequel we consider sets of square bricks with an effective alphabet of at least two symbols.

Also note that apart from the single-label case, the decidability of sets of squares does not depend on the size of the effective alphabet, since a larger alphabet can be "simulated" with two symbols at the expense of the size of the squares. When arbitrary shapes are considered, shapes can be used to simulate labels, thus making brick decidability equivalent to polyomino decidability.

3.1 The general case

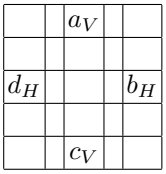
Theorem 3. *It is undecidable whether $X \subseteq A^{\bowtie}$ is a code.*

Proof. We construct a reduction from the Wang tilability problem which is known to be undecidable.

A *Wang tile* is a quadruple (a, b, c, d), where a, b, c, d belong to a given set of colours. Let T be a set of Wang tiles over the colours $C \cup \{\beta\}$, where β denotes a "blank" colour. Assume that T does not contain the blank tile, i.e., $(\beta, \beta, \beta, \beta)$. We construct a set of bricks X (over an appropriate alphabet) such that a finite tiling over T exists iff X is not a code.

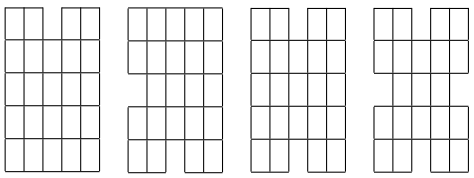
Take two copies of the colour set C: C_H ("horizontal colours") and C_V ("vertical colours"). For $c \in C$, c_H denotes the corresponding element of C_H, and c_V is the element of C_V. Define $A = C_H \cup C_V \cup \{\beta\}$.

Each tile $t = (a, b, c, d) \in T$ is mapped into a 5×5 brick t':

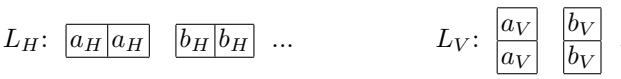


The set of all bricks resulting from the mapping of tiles in T is denoted by T'.

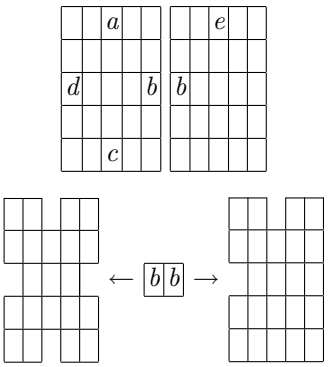
Now define "shape constructors" K. It contains all 5×5 bricks over $\{\beta\}$, with at least one of the edge mid-points removed, i.e., $K = \{s \in \{\beta\}^{\bowtie} \mid \operatorname{dom} s = \{0,1,2,3,4\}^2 \setminus P,\ P \in \mathcal{P}(\{(2,4),(4,2),(2,0),(0,2)\}),\ P \neq \emptyset\}$. For instance:



We still need "horizontal links", L_H, and "vertical links", L_V. Horizontal links are 1×2 dominoes labelled uniformly with all available horizontal colours. Similarly, vertical links are 2×1 dominoes labelled with vertical colours.



Define $X = T' \cup K \cup L_H \cup L_V$. Assume now that T admits a finite tiling. Consider two adjacent tiles, t_1 and t_2, with a common edge labelled with $c_{H/V}$. This can be modelled with the bricks in X in two ways: either $t'_1, t'_2 \in T'$ or two shape constructors with a $c_{H/V}$ link can be used. If a non-blank tile t is adjacent to a blank tile, t' has a blank edge which is modelled by a shape constructor with no "cave" at the corresponding edge. Blank tiles are not represented in the brick model, thus the model is finite. An example of two factorizations of adjacent tiles is shown below.



Conversely, if a brick admits two factorizations over X, it models some tiling over T. Thus, a finite tiling over T exists iff X is not a code. □

3.2 The case of two square bricks

We now consider sets consisting of just two bricks, each of them being a square (in the geometrical sense). We show that there exists a simple algorithm to verify whether a given set of this kind is a brick code.

Theorem 4. *Let $X = \{k, l\} \subseteq A^{\bowtie}$, where $\operatorname{dom} k$ and $\operatorname{dom} l$ are squares, $k \neq l$. Then X is not a brick code iff k and l have a common rectangular tiler, i.e., there exists a rectangle $t \in A^{\bowtie}$ such that $k, l \in \{t\}^{\bowtie}$.*

Proof. The "if" part is obvious. Now, if $|\operatorname{dom} k| = |\operatorname{dom} l|$, then k and l have identical shape and differ in their labelling, so X is obviously a code with no common tiler for k and l. Thus we are left with the case, e.g., $|\operatorname{dom} k| < |\operatorname{dom} l|$.

Assume that X is not a brick code. There exists $y \in X^{\bowtie}$ such that y admits two different tilings with the elements of X.

Take the leftmost cell in the uppermost row of y. If both tilings use the same square $x \in \{k, l\}$ to cover this cell, then remove x from y and repeat this procedure until y' is obtained such that the two tilings place different squares in the leftmost cell in the uppermost row of y'. In other words, we take y to be a minimal brick admitting two tilings over X. Call the tilings T_k and T_l, according to the square being used to cover the cell specified above.

Now this implies that k tiles the top-left corner of l. Moving rightwards along the top edge of l, we observe that when T_k covers the remaining part of l, it cannot place a tile higher than the top edge of l, since we have started in the uppermost row. Thus the next tile to the right of k has to be aligned along the top edge. Since we already know that l contains a copy of k in its top-left corner, we have another copy of k "along the way" (although l may have actually been used). Continuing in this way, we observe that the two tilings will eventually arrive at a common right edge when they reach the lowest common multiple of the edge lengths of k and l.

$$\boxed{k_1 \ \ldots \ k_n \ k_1 \ \ldots \ k_i} \, k_1 \quad k_2 \quad \ldots \ k_{n-i}$$
$$\boxed{k_1 \ \ldots \ k_n} \, k_1 \ \ldots \ k_i \ \ k_{i+1} \, k_{i+2} \, \ldots \ k_n$$

Considering the situation to the right of the first copy of l in T_l and denoting the columns of k by $k_1, k_2, ..., k_n$ we obtain $k_1 = k_{i+1}$, $k_2 = k_{i+2}$, ..., $k_{n-i} = k_n$ where $i = (\operatorname{len} l) \operatorname{mod} (\operatorname{len} k)$ (cf. figure above). If $i = 0$, then $\operatorname{len} k$ is the width of the common tiler. Otherwise this width is equal to i.

The above argument can be repeated in vertical direction, thus giving a square that can be tiled with k or with l. The size of the square will be the lowest common multiple of the sizes of k and l. □

Note that the proof becomes trivial if the effective alphabet of X is just one symbol, since X is never a code then with, e.g., the unit square being the common tiler for k and l

Example 6. Consider $X = \{k, l\}$ containing two bricks depicted below. They have a common tiler t, hence X is not a code.

$$\boxed{\begin{array}{cccc} a & b & a & b \\ b & b & b & b \\ a & b & a & b \\ b & b & b & b \end{array}} \qquad \boxed{\begin{array}{cccccc} a & b & a & b & a & b \\ b & b & b & b & b & b \\ a & b & a & b & a & b \\ b & b & b & b & b & b \\ a & b & a & b & a & b \\ b & b & b & b & b & b \end{array}} \qquad t = \boxed{\begin{array}{cc} a & b \\ b & b \end{array}}$$

Proposition 7. *If $k, l \in A^{\bowtie}$ have a common rectangular tiler, then they have a common square tiler.*

Proof. Assume that $k, l \in \{t\}^{\bowtie}$, where t is a rectangle of size $p \times q$. Let r be the lowest common multiple of p and q. Both $\operatorname{len} k$ and $\operatorname{len} l$ have to be multiples of r. Hence, $t^{(r/p)\times(r/q)}$ can be taken as the common square tiler. □

Corollary 1. *Let $X = \{k, l\} \subseteq A^{\bowtie}$, where $\operatorname{dom} k$ and $\operatorname{dom} l$ are squares. It is decidable whether X is a brick code.*

3.3 The case of 15 square bricks

We show that a Thue system can be reduced to a brick code problem with square bricks. Choosing a small Thue system with an undecidable word problem, we obtain a set of 15 squares, thus proving that the codicity of a set containing 15 square bricks is undecidable.

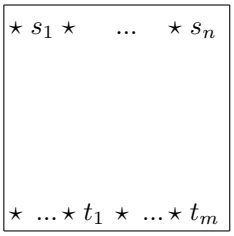
There exists a non-erasing Thue system with an undecidable word problem over a two-letter alphabet with just three relations. This is the smallest example known to us, due to Matiyasevich [8, 9]. We can encode this system, including

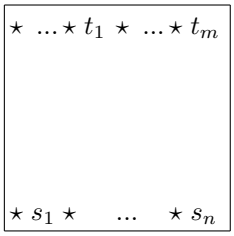
two arbitrary input words v and w, in a set X containing 15 square bricks and having the following property: v and w are equivalent (in the Thue sense) iff X is not a brick code. This implies undecidability of the codicity problem for 15 squares.

Consider a finite alphabet Σ and a finite subset $S \subseteq \Sigma^* \times \Sigma^*$. The elements of S are called relations. A *Thue system* is the pair (Σ, S). For $u, v \in \Sigma^*$, we write $u \sim_S v$ if there exists a relation $(s,t) \in S$ or $(t,s) \in S$ such that $u = xsy$ and $v = xty$ for some $x, y \in \Sigma^*$. By $\equiv_S$ we denote the transitive closure of $\sim_S$. The word problem is: given $u, v \in \Sigma^*$, does $u \equiv_S v$ hold?

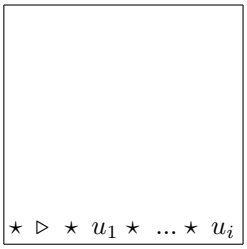
For a given Thue system (Σ, S) and two words $u, v \in \Sigma^*$ we construct a set $X_{S,u,v}$ of square bricks over an alphabet $A = \Sigma \cup \{\star, \triangleright\}$ in the way shown below. Cells which are left blank in the diagrams should be labelled with $\star$.

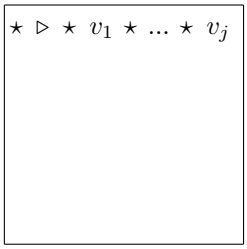
– For each relation $(s_1...s_n, t_1...t_m) \in S$ we draw two bricks. Note that if one of the words is shorter, it is left-padded with $\star$ symbols:



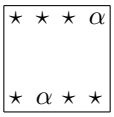


– For the two words $u = u_1...u_i$ and $v = v_1...v_j$ we draw:





– For each symbol $\alpha \in \Sigma \cup \{\triangleright\}$ we draw ("rewriting bricks"):



– Finally, we draw an additional square (a "filler"): 

Theorem 5. *Let (Σ, S) be a Thue system and let $u, v \in \Sigma^*$. Let $X_{S,u,v}$ be the set constructed as described above. The following equivalence holds: $u \equiv_S v$ iff $X_{S,u,v}$ is not a brick code.*

Proof. ($\Rightarrow$) We first present a sketch of the proof, and then give the details.

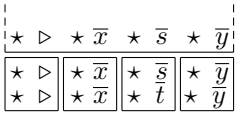
A successful derivation in the Thue system corresponds to a series of bricks, stacked vertically in layers: one of the input words, bricks corresponding to the first derivation step, bricks corresponding to the second step, ..., the other input word. Adjacent bricks have the same labels along level boundaries. Rewriting bricks are used to shift non-$\star$ symbols to the left and to rewrite symbols that are not changed at each step.

Now note that a tiling described above can also be constructed using the symbol-rewriting bricks and the filler. This implies non-codicity of the set.

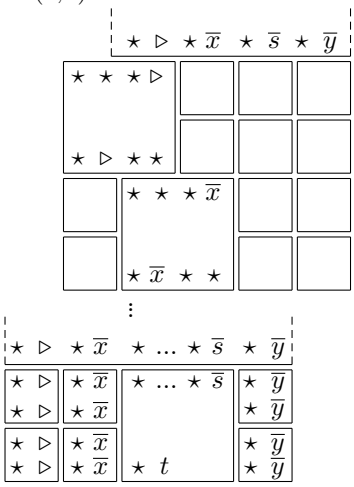
To analyze the details of this construction, consider $u \equiv_S v$. Let $\overline{\omega}$ denote the sequence of symbols $\omega_1 \star \omega_2 \star ... \star \omega_n$, where $\omega = \omega_1\omega_2...\omega_n \in \Sigma^*$. Using the Thue derivation, we construct a shape as follows:

1. Initial lines are the brick corresponding to u:

2. If the derivation includes $u' \sim_S u''$, there exists a relation (s, t) such that $u' = xsy$ and $u'' = xty$. Consider the cases:
   - If $|s| = |t|$, a brick is added as follows:



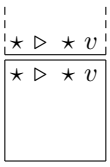
   - If $|s| < |t|$, $\star$ symbols have to be added before a brick that corresponds to (s, t):



– If $|s| > |t|$, a brick is added in a similar way but $\star$ symbols are removed afterwards.

Note that in each case after a brick corresponding to (s, t) is added, the bottom row corresponds to u''.

3. If the bottom row corresponds to v, the construction is complete:



Another tiling for the shape being constructed can be made with the rewriting bricks and the filler (cf. Example 7).

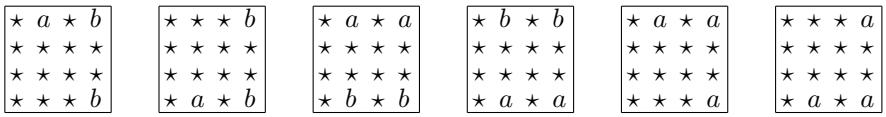
($\Leftarrow$) If $u \equiv_S v$ does not hold, there is no word u' such that $u' \sim_S v$, and consequently it is not possible the complete the construction as in case 3. □

Note that the reduction works even if the Thue system is erasing. A semi-Thue system could also be used with a minor modification (no symmetrical relation bricks). Matiyasevich and Sénizergues give an example of a semi-Thue system over a three-letter alphabet with three relations that has, e.g., un undecidable individual accessibility problem (see [10]). This leads to a similar result.

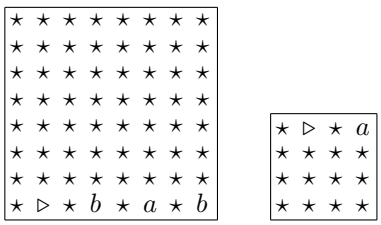
Example 7. The Thue system of Matiyasevich contains a relation of considerable length; thus it is not well-suited as an example. To clarify the idea of the Thue-to-brick reduction, we present a Thue system with short relations. Let $S = \{(ab, b), (aa, bb), (aa, a)\}$. We ask whether $bab \equiv_S a$. The obvious answer is yes, thus the set of bricks X for this Thue system is not a code.

The set X consists of the following bricks:

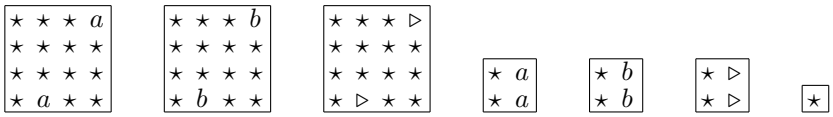
– bricks corresponding to the rules:



– bricks corresponding to the input words:



– rewriting bricks and the filler:



The following derivation corresponds to a tiling shown in Fig. 1. Underlined subwords are the ones being substituted at each step.

$$b\underline{ab} \sim_S \underline{bb} \sim_S \underline{aa} \sim_S a$$

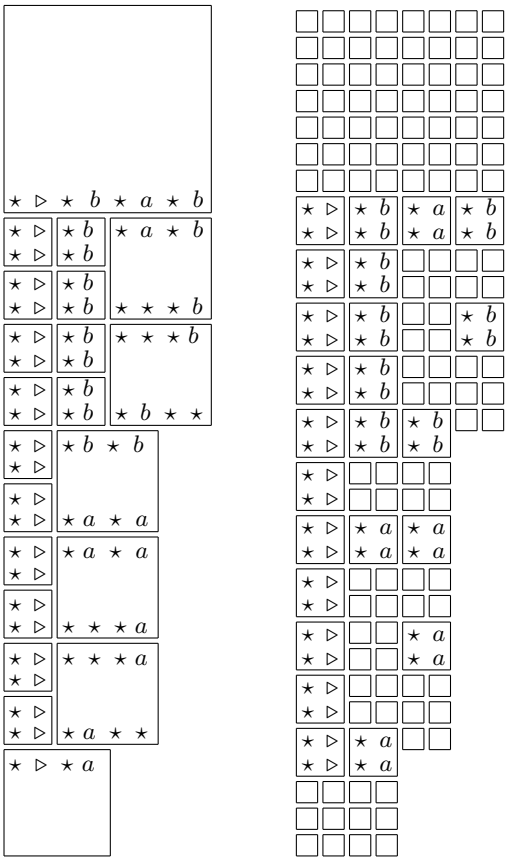


Fig. 1. Tilings corresponding to $b\underline{ab} \sim_S \underline{bb} \sim_S \underline{aa} \sim_S a$

Corollary 2. *The codicity problem for sets containing 15 or more square bricks is undecidable.*

Proof. The reduction of a Thue system with three relations over a two-letter alphabet produces six bricks corresponding to the relations, two corresponding to the input words, six rewriting ones, and the filler. □

References

1. P. Aigrain, D. Beauquier: Polyomino tilings, cellular automata and codicity. Theoret. Comp. Sci. **147** (1995) 165–180
2. D. Beauquier, M. Nivat: A codicity undecidable problem in the plane. Theoret. Comp. Sci. **303** (2003) 417–430
3. J. Berstel, D. Perrin: *Theory of Codes.* Academic Press (1985)
4. S.W. Golomb: *Polyominoes.* Princeton University Press, 2nd edition (1996)
5. M. Ito, H. Jürgensen, H.J. Shyr, G. Thierrin: Outfix and Infix Codes and Related Classes of Languages. Journal of Computer and System Sciences **43** (1991)
6. J. Karhumäki: Some open problems in combinatorics of words and related areas. TUCS Technical Report **359** (2000)
7. M. Margenstern: Frontiers between decidability and undecidability: a survey. Theoret. Comp. Sci. **231** (2000) 217–251
8. Yu. Matiyasevich: Simple examples of undecidable associative calculi. Soviet Math. Dokladi **8** (1967) 555–557
9. Yu. Matiyasevich: Word problem for Thue systems with a few relations. Lecture Notes in Computer Science **909** (1995) 39–53
10. Yu. Matiyasevich, G. Sénizergues: Decision problems for semi-Thue systems with a few rules. Proceedings LICS'96 (1996) 523–531
11. M. Moczurad, W. Moczurad: Decidability of simple brick codes. In: *Mathematics and Computer Science III (Algorithms, Trees, Combinatorics and Probabilities).* Trends in Mathematics, Birkhäuser (2004)
12. M. Moczurad, W. Moczurad: Some open problems in decidability of brick (labelled polyomino) codes. Cocoon 2004 Proceedings, Lecture Notes in Computer Science **3106** (2004) 72–81
13. W. Moczurad: Brick Codes. In: C.L. Nehaniv and M. Ito (eds.), *Algebraic Engineering.* World Scientific Publishers (1999)
14. W. Moczurad: Algebraic and algorithmic properties of brick codes. Doctoral thesis, Jagiellonian University (1999)
15. W. Moczurad: Brick codes: families, properties, relations. Intern. J. Comp. Math. **74** (2000) 133–150
16. M.P. Schützenberger: Une théorie algébrique du codage. Séminaire Dubreil-Pisot (1955)
17. H.J. Shyr: *Free Monoids and Languages.* Hon Min Book Company, Taiwan (1991)