Shape Recognition by a Finite Automaton Robot *

Robert Gmyr[1], Kristian Hinnenthal[1], Irina Kostitsyna[2], Fabian Kuhn[3], Dorian Rudolph[1], and Christian Scheideler[4]

1 **Paderborn University, Germany**
{gmyr, krijan, dorian}@mail.upb.de
2 **TU Eindhoven, the Netherlands**
i.kostitsyna@tue.nl
3 **University of Freiburg, Germany**
kuhn@cs.uni-freiburg.de
4 **Paderborn University, Germany**
scheideler@upb.de

Abstract

We investigate the problem of recognizing the ratio of the sides of a parallelogram by a finite-state automaton robot with pebbles operating on a triangular grid. Our goal is to determine the computational power of a single finite automaton robot in this setting with and without the help of pebbles. We demonstrate that a robot without pebbles can determine whether a given shape is a parallelogram whose sides have a ratio of h to $ah+b$ for constant integers a and b, but cannot detect whether the ratio is h to $f(h)$, where $f(x) = \omega(x)$ is a superlinear function. For a robot with a single pebble, we present an algorithm to decide whether the sides ratio is h to $p(h)$ for a given polynomial $p(\cdot)$ of constant degree. Finally, we present algorithms to decide more complex functions, such as exponential functions, using multiple pebbles.

1 Introduction

Motivated by the problem of shape recognition in restricted computational models, we study the problem of recognizing shapes of specific side ratios. We build upon the model introduced in [7] where finite-state automaton robots move on a triangular grid and assemble hexagonal tiles into various shapes.

In this paper, rather than exploring shape formation problems, we investigate the problem of *shape recognition* by a single robot. Specifically, we begin with testing whether a given tile formation is of a certain simple shape—in particular, a parallelogram—and then deciding for a given function f whether the longer side has length $f(h)$ where h is the shorter side's length. We further consider a variant of this problem where the robot is given a set of pebbles that can be used to mark certain positions. Our ultimate goal is to investigate the computational capabilities of a simple robot concerning shape recognition, and to what extent the robot can benefit from employing pebbles.

Model. Let a single *robot* be placed on a finite set of *hexagonal tiles*; each tile occupies exactly one node of an infinite triangular grid graph, as shown in Figure 1, such that the subgraph induced by all nodes occupied by tiles is connected. The robot is initially placed on a tile and carries a (possibly empty) set of *pebbles*. It can drop a pebble on a tile that is not already occupied by another pebble.

The robot acts as a *deterministic finite automaton* and operates in *look-compute-move* cycles. In the *look* phase the robot can observe the node it occupies and the six neighbors

* This work was partially supported by DFG grant SCHE 1592/3-1. Fabian Kuhn is supported by ERC Grant No. 336495 (ACDC).

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

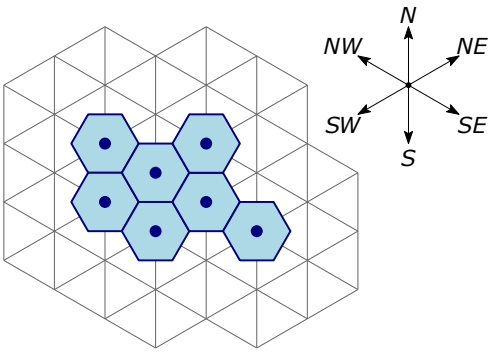

Figure 1 An exemplary tile configuration. The top right part of the figure shows the compass directions we use to describe the movement of a robot.

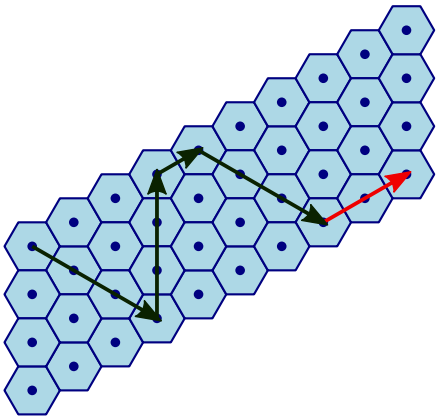

Figure 2 A N-NE-parallelogram with height 4 and length $10 = 2 \cdot 4 + 2$. The black arrows indicate the zig-zag movement of a robot as described in the proof of Theorem 3.1. The red arrow shows the final NE movement.

of that node. For each of these nodes it can determine whether it is occupied by a tile and whether a pebble is placed on that tile. In the *compute* phase the robot potentially changes its state and determines its next move according to the observed information and the number of carried pebbles. In the *move* phase the robot can either take a pebble from its current node, place a pebble it is carrying at that node, or move to an adjacent tile.

Note that even though we describe the algorithms as if the robot knew its global orientation, we do not actually require the robot to have a compass. For the algorithms presented in this paper, it is enough for the robot to be able to maintain its relative orientation with respect to its original orientation.

Related Work. To the best of our knowledge, shape recognition has never been investigated in our model. However, solving problems by traversing a tile structure with simple agents has been studied in many different areas. For instance, [14] considers the problem of deciding whether a structure is simply-connected. Other problems include Gathering and Rendezvous (e.g., [13]), Intruder Caption and Graph Searching (e.g., [2], [5]), or Black Hole Search (e.g., [12]).

For many of the above problems it has also been investigated whether pebbles can be helpful. This question is particularly well-studied for the classical Network Exploration Problem (see, e.g., [4]). For example, it is known that a finite automaton robot can neither explore all planar graphs [6] nor find its way out of a planar labyrinth [3] without any pebble. For the Labyrinth Exploration Problem (see [10, 11] for a comprehensive survey), it is known that having an additional pebble does not help the robot [8]. However, a robot with two pebbles can solve the problem [1].

2 Recognizing Simple Shapes

First, observe that a single robot can easily detect whether the initial structure is a line, a triangle, a hexagon, or a parallelogram.

Line. For example, to test if a given tile shape is a line, the robot first chooses a direction in which there is a tile (say, w.l.o.g., N), walks in that direction as far as possible (i.e., until

there is no tile in that direction anymore), and then traverses the structure into the opposite direction until no longer possible. If it ever encounters a tile to the left or right of any traversed tile, the structure is not a line.

Parallelogram. To test if a given tile shape is a (filled) parallelogram axis aligned along the directions N and NE (see Figure 2), first, the robot moves to a locally southernmost tile of the structure by moving S and SW as long as there is a tile in any of these directions. It then traverses the shape column by column in a snake-like fashion by repeating the following movements: First, the robot moves N as far as possible; it then moves one step NE; then it moves S as far as possible, and it finally moves one step NE. The above procedure is repeated until a NE movement is impossible. By performing local checks alongside the movements described above the robot can verify whether the tile shape is a parallelogram.

The other simple shapes can be easily tested in a similar fashion.

▶ Observation 2.1. A robot without any pebble can detect whether the initial tile configuration is a line, a triangle, a hexagon, or a parallelogram.

3 Recognizing Parallelograms with Specific Side Ratio

As noted in Observation 2.1, a single robot without pebbles can verify whether a given shape is a parallelogram. To investigate the computational power of a finite automaton, in this section we consider the problem of deciding whether a parallelogram has a given side ratio. Additionally, we examine how pebbles can be helpful to decide more complex side ratios.

We assume w.l.o.g. that the robot needs to detect whether the given tile configuration is a parallelogram that is axis-aligned along the north and north-east direction (we say *N-NE-parallelogram*), as indicated in Figure 2. We denote a maximal sequence of consecutive tiles from N to S as a *column* and a maximal sequence of consecutive tiles from SW to NE as a *row*. Let h be the size of each column, i.e., the parallelogram's *height*, ℓ be the size of each row, i.e., the parallelogram's *length*, and let $h \leq \ell$. We number the columns of the parallelogram from 0 to $\ell - 1$ growing in the north-eastern direction.

3.1 A Robot without any Pebble

First, we point out that a single robot can detect whether the structure is a parallelogram in which its length ℓ is a linear function of its height h.

▶ **Theorem 3.1.** *A single robot can detect whether the tile configuration is a parallelogram with $\ell = ah + b$ for any constants $a, b \in \mathbb{N}$.*

Proof. First, the robot verifies whether the structure is a parallelogram. If so, the robot moves to the northernmost tile of the column 0. It then traverses the tile structure in two stages to verify the ratio of the sides. In the first stage the robot "measures" the distance ah along the length of the parallelogram moving in a zig-zag fashion as depicted in Figure 2. In the second stage the robot measures the second term b. More specifically, in the first stage, the robot repeats the following movements in a loop: (1) move SE as far as possible, (2) move N as far as possible, and (3) make one step NE. After having performed the complete sequence of SE movements a times, the robot moves on to the second stage, in which it makes an additional b NE steps.

If the robot reaches the easternmost column before completing the above procedure, or finally halts on a tile with a neighboring tile at NE, it terminates with a negative result. Otherwise, it terminates with a positive result. It is easy to see that $\ell = ah + b$ if and only if the robot terminates with a positive result. ◀

▶ Remark. The algorithm in the previous theorem can be adapted for $b \in \mathbb{Z}$, i.e., b can also be a negative integer. To achieve that, we halt the execution of the first stage once the robot has performed $|b|$ *SE* steps, then move *SW* as far as possible, and continue the first stage from there. Then, $\ell = ah + b$ if and only if the robot eventually reaches the southernmost tile of the easternmost column.

▶ Remark. The algorithm can be further extended to apply in the case of a rational a. Let $a = p/q$ be an irreducible fraction. Instead of moving in a zig-zag fashion in the first stage, the robot alternates between moving p steps *NE* and q steps *S*. To exactly end up at the southernmost tile of the easternmost column, the robot needs to skip the very first *NE* and *S* step.

We have shown that a single robot can determine whether the length of a parallelogram is given by a certain linear function of its height. However, that is as much as one robot can hope for. Indeed, one robot is not able to decide whether the length of the parallelogram is given by a superlinear function of its height, as the following theorem states. Its proof can be found in the full version of this paper.

▶ **Theorem 3.2.** *A single robot without any pebbles cannot decide whether the tile configuration is a parallelogram with $\ell = f(h)$, where $f(x) = \omega(x)$.*

3.2 A Robot with a Single Pebble

In the following, we demonstrate that, in contrast to the negative result of Theorem 3.2, a single robot can decide any polynomial of constant degree.

▶ **Theorem 3.3.** *A single robot with a pebble can decide whether the tile configuration is a parallelogram of height h and length $\ell = p(h)$ for any given polynomial $p(\cdot)$ of constant degree n.*

Proof. Define the *falling factorial* of x as $(x)_i := x(x-1)\cdots(x-i+1)$, and transform the input polynomial into the form $p(x) = a_n \cdot (x)_n + a_{n-1} \cdot (x)_{n-1} + \ldots + a_0$. We will show that the robot can move the pebble in phases, by $|a_i \cdot (h)_i|$ steps in each phase i. Let $\mathrm{lcm}_i(x) := \mathrm{lcm}(x, \ldots, x-i+1)$, where lcm is the *least common multiple*, and $g_i(x) := (x)_i / \mathrm{lcm}_i(x)$. From [9] it follows that $\mathrm{lcm}_i(x) \mid (x)_i$, and that $g_i(x)$ is periodic with period $\mathrm{lcm}(1, \ldots, i-1)$, i.e., $g_i(x) = g_i(x + \mathrm{lcm}(1, \ldots, i-1))$. Let $p_i(x)$ be the sum of the first $n-i$ summands of $p(x)$, i.e., $p_i(x) = a_n \cdot (x)_n + a_{n-1} \cdot (x)_{n-1} + \ldots + a_{n-i+1} \cdot (x)_{n-i+1}$.

Initially, the pebble is located on the northernmost tile of column 0. To test whether $\ell = p(h)$, the robot will move the pebble along the northernmost row in phases, until eventually it is shifted $p(h) - 1$ steps to the *NE* from its original position. If upon termination the pebble is located at the northernmost tile of column $\ell - 1$, then $p(h) = \ell$.

The algorithm proceeds in phases $n, \ldots, 0$. We maintain the invariant that after phase i for all $i > 0$, the pebble is located at the northernmost tile of column $p_i(h)$. That is, in phase i, the robot moves the pebble $|a_i \cdot (h)_i|$ steps *NE*, if a_i is positive, and *SW*, otherwise. In the final phase $i = 0$, the robot moves the pebble by $|a_0 - 1|$ steps *NE*, if $a_0 \geq 1$, and *SW*, otherwise. For now, assume that each movement can be carried out without moving the pebble outside of the parallelogram. We will later describe how to lift this restriction.

We now describe how the pebble is moved by $|a_i \cdot (h)_i|$ steps. First, note that $a_i \cdot (h)_i = a_i \cdot g_i(h) \cdot \mathrm{lcm}_i(h)$. The first factor a_i is a constant. The second factor $g_i(h)$ can be determined as follows. We preliminarily encode all possible values of $g_i(\cdot)$ for all $i \in \{0, \ldots, n\}$ into the robot's memory, which can be done since n is constant and $g_i(\cdot)$ has a constant period. Before the main algorithm's execution, the robot can compute $g_i(h)$ for all i by moving

through the westernmost column from north to south: Starting with $g_i(1)$, in every step to the south the robot computes the subsequent function value until the period of $g_i(\cdot)$ is reached, in which case it restarts with $g_i(1)$. When it reaches the southernmost tile of the column, it knows $g_i(h)$ for all i.

We next show how the robot moves the pebble by $\text{lcm}_i(h)$ steps, which, by repeating the movement $|a_i \cdot g_i(h)|$ times, concludes how the complete movement by $|a_i \cdot (h)_i|$ steps is performed. Assume the pebble is in some column c and $\text{lcm}_i(h) \mid c$ (which we will prove by induction shortly). The robot alternates between the following two operations: (1) move the pebble into column c' by moving it one step *NE*, if $a_i > 0$, or *SE*, otherwise; (2) verify whether $\text{lcm}_i(h) \mid c'$ as follows. The robot first performs the zig-zag movement from the proof of Theorem 3.1 to verify whether $h \mid c'$, i.e., whether a *NE* movement moves the robot onto a tile occupied by the pebble. It continues to analogously verify whether $h-1 \mid c'$, $h-2 \mid c'$, $\ldots$, $h-i+1 \mid c'$ by performing a modified zig-zag movement an additional $i-1$ times. Here, the zig-zags of the j-th verification are adjusted accordingly by moving j steps S prior to each sequence of *SE* movements. The robot stops alternating between the two above operations once the pebble has been moved to a column c' such that $\text{lcm}_i(h) \mid c'$ for the first time. Then, the pebble must have been moved by $\text{lcm}_i(h)$ steps.

It remains to prove that when the robot wants to move the pebble, currently occupying a node of column c, by $\text{lcm}_i(h)$ steps for some i, then $\text{lcm}_i(h) \mid c$. The invariant holds initially for $c = 0$. Now assume it holds immediately after having moved the pebble by $\text{lcm}_i(h)$ steps into column $c \pm \text{lcm}_i(h)$. Afterwards, the robot can move it by either $\text{lcm}_i(h)$ steps again, in which case $\text{lcm}_i(h) \mid c \pm \text{lcm}_i(h)$ holds, or it wants to move it by $\text{lcm}_{i-1}(h)$ steps, and $\text{lcm}_{i-1}(h) \mid c \pm \text{lcm}_i(h)$ holds since $\text{lcm}_{i-1}(h) \mid \text{lcm}_i(h)$.

Finally, we show how the robot can resolve *overflows*, i.e., situations in which the above algorithm would move the pebble outside of the parallelogram. First, note that the execution of the algorithm after an overflow can, in principle, be continued by the robot by "mirroring" all movements beyond the westernmost or easternmost column, carrying them out into reverse direction. Assume that h is sufficiently large such that $\mid a_i \cdot (h)_i + \ldots + a_0 \mid \leq p(h)$ for all i. For all small $h = O(\max_i(|a_i|))$ we can encode the constantly many possible function values into the robot's state and test them prior to the algorithm's execution by traversing the two sides of the parallelogram once. If throughout the execution of the algorithm the robot ever attempts to move the pebble into a column west of column 0 or east of "virtual" column 2ℓ (while performing the mirroring method from above), it would subsequently not be able to ever move the pebble back into column ℓ (following from the assumption that h is sufficiently large), and consequently $\ell \neq p(h)$. Therefore, the robot can prematurely terminate with a negative result whenever it encounters such a situation. ◀

In contrast to the previous theorem, for a fixed k, we believe that a polynomial $p(\cdot)$ of degree $\omega(k)$ can be constructed such that a single robot with k states and a pebble cannot decide whether $\ell = p(h)$.

3.3 A Robot with Multiple Pebbles

Finally, we show that having multiple pebbles enables the robot to decide some more complex functions.

▶ **Theorem 3.4.** *A robot with two pebbles can decide whether the tile configuration is a parallelogram with $\ell = 2^h$.*

Proof. We only provide a high-level idea of the algorithm and omit some of the implementation details. We call the pebbles a and b. Pebble a will always be in the northernmost row of

the parallelogram. The robot first places pebble a at the northernmost tile of column 1, and b on the southern neighbor of a. It then repeatedly performs the following procedure: First, the robot alternatingly moves a *NE* and b *SW* until there is no tile *SW* from b. Afterwards, the robot brings b back into the current column of a by repeatedly moving b *NE* and verifying whether a is in the northernmost tile of the column. It then moves b one step *S*.

In the i-th iteration of the above procedure, pebble a gets moved 2^{i-1} times. Thus, the distance between a and the westernmost column doubles in each iteration. By moving b one step *S*, the robot counts the number of doubling operations up to h. If a has reached the easternmost column after h doubling operations, then $\ell = 2^h$. ◀

The proof of the following theorem can be found in the full version of this paper.

▶ **Theorem 3.5.** *A robot with three pebbles can decide whether a given tile configuration is a parallelogram with*

$$\ell = 2^{2^{\cdot^{\cdot^{\cdot^{2^h}}}}},$$

where the power tower is of constant height.

▶ Remark. It can be shown that a single robot can also decide a power tower of height h.

References

1 M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Proc. 19th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 132–142, 1978.
2 A. Bonato and R. J. Nowakowski. *The Game of Cops and Robbers on Graphs*. AMS, 2011.
3 L. Budach. Automata and labyrinths. *Mathematische Nachrichten*, 86(1):195–282, 1978.
4 S. Das. Mobile agents in distributed computing: Network exploration. *Bulletin of the EATCS*, 109:54–69, 2013.
5 F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245, jun 2008.
6 P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2-3):331–344, 2005.
7 R. Gmyr, I. Kostitsyna, F. Kuhn, C. Scheideler, and T. Strothmann. Forming tile shapes with a single robot. In *Abstr. European Workshop on Computational Geometry (EuroCG)*, pages 9–12, 2017.
8 F. Hoffmann. One pebble does not suffice to search plane labyrinths. In *Proc. International Fundamentals of Computation Theory Conference (FCT)*, pages 433–444, 1981.
9 S. Hong and Y. Yang. On the periodicity of an arithmetical function. *Comptes Rendus Mathematique*, 346(13):717–721, 2008.
10 G. Kilibarda, V. B. Kudryavtsev, and Š. Uščumlić. Collectives of automata in labyrinths. *Discrete Mathematics and Applications*, 13(5):429–466, 2003.
11 G. Kilibarda, V. B. Kudryavtsev, and Š. Uščumlić. Independent systems of automata in labyrinths. *Discrete Mathematics and Applications*, 13(3):221–225, 2003.
12 E. Markou. Identifying hostile nodes in networks using mobile agents. *Bulletin of the EATCS*, 108:93–129, 2012.
13 A. Pelc. Deterministic rendezvous in networks: A comprehensive survey. *Networks*, 59(3):331–347, 2012.
14 A. N. Shah. Pebble automata on arrays. *Computer Graphics and Image Processing*, 3(3):236–246, 1974.