INFO-F420 Project report
Undecidability of the 5-polyomino tiling problem

Camille Leduc

November 2024

1 Introduction

This project builds upon Yang and Zhang paper on the undecidability of the tiling problem using a set of 8 polyominoes [9]. The details of their paper will be discussed further in Section 3. The focus of this project is to investigate whether the tiling problem remains undecidable with a set of 5 polyominoes.

Therefore, this report will first outline key concepts introduced in Yang and Zhang's paper. Secondly, the paper will be discussed as mentioned before. Finally, the methodology employed to address Wang domino problem, followed by a discussion of its implementation.

2 Key concepts

2.1 Wang tiles

A Wang tile is a unit square with each edge assigned a color, a set of 3 tiles is illustrated in Figure 1. Wang's domino problem involves tiling the entire plane with translated copies from a finite set of Wang tiles. The tiles must be edge-to-edge and the color of common edges between two adjacent Wang tiles must be the same [9].

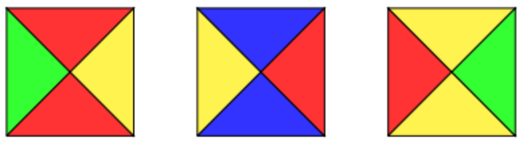


Figure 1: A set of 3 Wang tiles [9]

Berger showed that Wang's domino problem is undecidable combining the facts of the existence of an aperiodic set of Wang tiles and the ability to simulate a Turing machine with Wang tiles [9][1].

Theorem 1. *Wang's domino problem is undecidable. [1]*

2.2 Polyominoes

The term polyominoes refers to n-ominoes with a variable n, where n represents the number of connected square units [2]. As you can see in Figure 2, a 5-omino, also called a pentomino, can have a multitude of shapes.

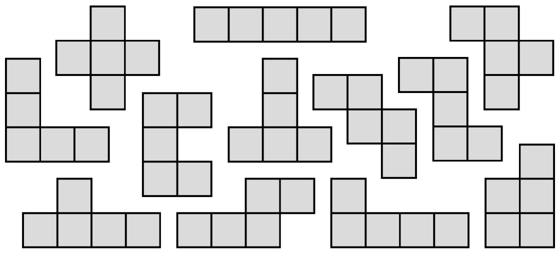


Figure 2: Set of pentominoes [6]

2.3 Turing machine

A Turing machine is a formal definition of computation device [4]. It takes an input over a given alphabet and outputs the result after several steps. The machine is based on three essential elements: a tape, a head and an internal state.

The tape is an infinite array of cells. Each cell contains a symbol and the input is put on the tape. The head marks a cell of the tape. The internal state of the machine and the symbol read at the position of the head allow the machine to make a decision at each step. This information is used to rewrite the symbol on the tape, change the internal state and shift the position of the head.

A Turing machine is formally defined by:

- A finite set of states Q with an accepting and a refusing state.
- An input alphabet Σ.
- A work alphabet $T \supset \Sigma$, containing the symbol '#', called the blank symbol.
- A transition function $\delta : T \times Q \to T \times Q \times \{-1, 1\}$.

The transition function such that $\delta(a, q) = (b, q', d)$ means that for a symbol a read at the head and the current internal state q, the symbol b will be written instead of a, the internal state becomes q' and the head shifts in the direction d. In Figure 3, you can see an automata representing the Turing machine.

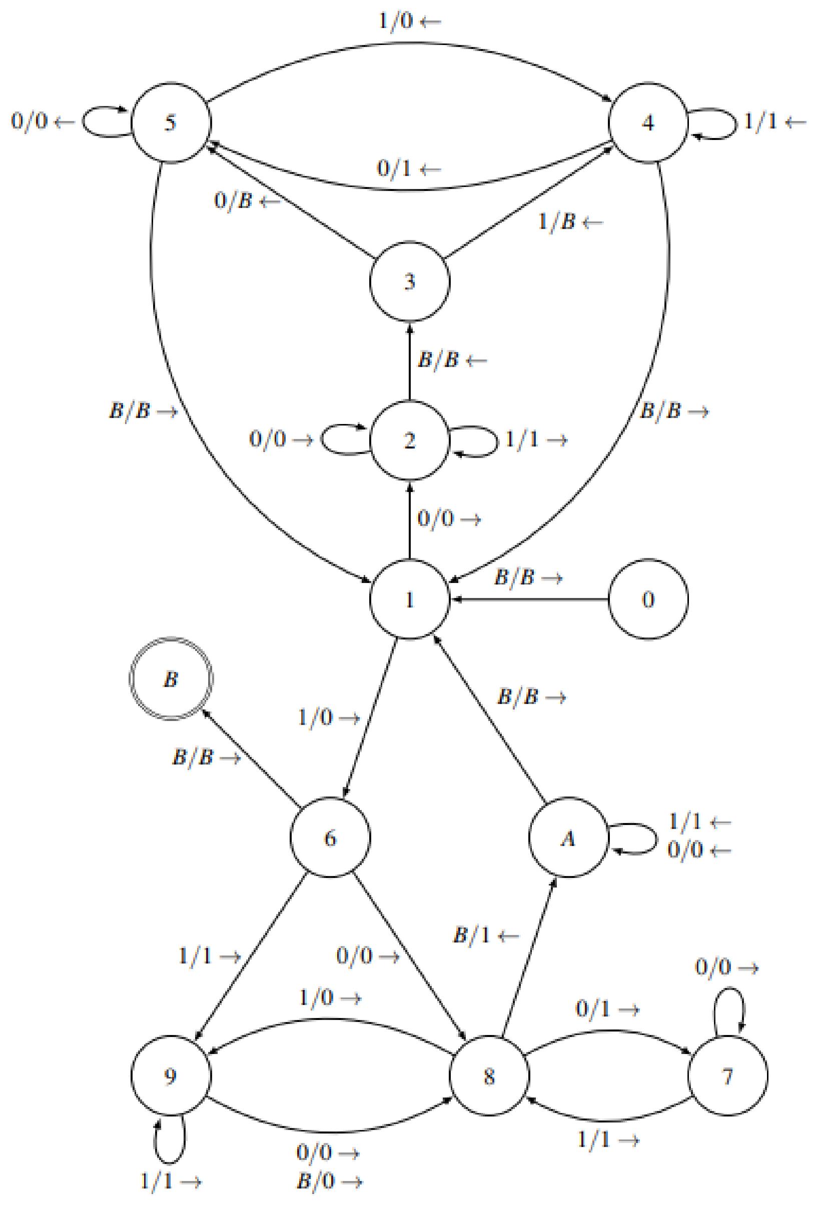


Figure 3: Collatz sequence Turing machine [4]

2.4 Undecidability

An undecidable problem is a problem that cannot be solved algorithmically. It is a decision problem for which there is no algorithm or Turing machine that can provide a yes or no answer. They must also have a stopping criterion such that for some inputs, the computation may run forever. To prove that a problem is undecidable, a known undecidable problem can be reduced to it. One of those undecidable problems is the Halting problem.

Theorem 2. *(Halting problem) There is no algorithm that, given a Turing machine M and a word u, decides if M halts on input u. [4]*

3 The paper: A proof of Ollinger's conjecture

Yang and Zhang focus on the undecidability of tiling the plane with a set of 8 polyominoes [9]. To do so, they use a reduction of Wang's domino problem towards the problem of tiling with a set of polyominoes, which was shown by Golomb [3]. The colors of the Wang tiles are simulated by bumps and dents on the each edge of a large square polyomino. Ollinger studied the undecidability with a set of fixed number of polyominoes and proved the following result.

Theorem 3. *The 11-polyomino tiling problem is undecidable. [5]*

Ollinger's conjecture is based on the existence of a set of 8 polyominoes. The k-polyomino tiling problem is undecidable for $k = 8, 9, 10$. They prove that the 8-polyomino tiling problem is undecidable. We will address their suggestion concerning the k-polyomino tiling problem for $2 \leq k \leq 7$.

4 Methodology

To prove that the tiling of the plane with Wang tiles is undecidable, we need to do a reduction. Following Theorem 4, we decide to reduce the Halting problem (see Section 2.4) to Wang's domino problem.

Theorem 4. *If there exists a reduction from a problem A to a problem B and problem A is known to be undecidable, then the problem B must also be undecidable.*

We use Jeandel and Vanier's Turing machine [4], which simulates the Collatz sequence (see Figure 3). It has a final accepting state and contains loops in certain states. On one hand, if the tiles build on the transition function of the Turing machine can tile the plane infinitely than the machine should never stop. On the other hand, the Turing machine should stop if the tiles cannot cover the plane infinitely.

5 Implementation

Three entities were build: Wang tiles, a Turing machine and a reduction from a Turing machine to Wang tiles.

5.1 Wang tiles

The `WangTile` object is a simple class. It has 4 attributes: north, east, south and west that represent the color of each edge. It also has a `draw` function that draws the color of each edge and its value when required.

A small interactive was created (see Figure 4). It allows the user to generate sets of 5 random Wang tiles.

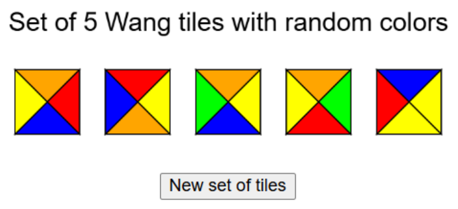


Figure 4: Wang tiles simulation

5.2 Turing machine

The `TuringMachine` class has the attributes mentioned in Section 2.3 as well as a `steps` attribute that stores the states of the machine after each step is executed. The previous and new states, the value read, the head, the tape and the direction are stored in a `MachineState` object.

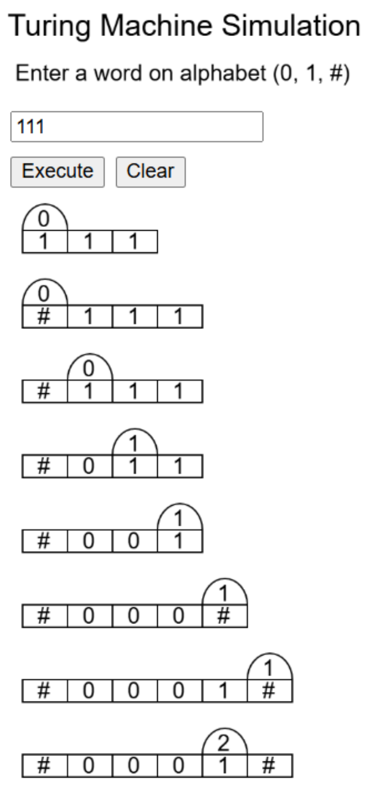
The method `run` takes the input as the machine tape and reads the tape while the final state is not reached. The method `read` gets the next step from the transition function based on the machine current state and the value read on the tape at the head position. The step contains b, q', d in a different data structure called `MachineStep`.

An interface allows the user to enter a word and see the execution steps of the Turing machine (see Figure 5).

5.3 Reduction

The `Reduction` class uses both the `WangTile` and `TuringMachine` classes. It has a machine as an attribute and it has the different tiles created from the transition table. There are 4 types of tiles, that you can see in Figure 6:

- Alphabet tiles that represent a letter of the alphabet with only the north and south edges having a value and color.
- Head tiles are used only for the first step and represents the head depending on the symbol on the tape on the south edge.
- Action tiles have the current state and the symbol read on the north edge, the new state on either the east or the west edge depending on the direction of the head and the symbol to write on the south edge.


Turing Machine Simulation

Enter a word on alphabet (0, 1, #)

```
111
```

Figure 5: Machine simulation for the word ”111”

- Move tiles are used to describe the movement of the head. It has the symbol of the tape at the new head location on the north edge, the current state (new state) on the opposite side edge of the action tile and the current state and the symbol on the south edge.

The action and move tiles represent together the transition of the head in one direction or the other.

The tiles are created at the construction of the reduction. The reduction runs by running its machine attribute and getting the execution steps of the machine. From this, we can create the step's equivalent in Wang tiles. The result is displayed in Figure 7.

6 Additional work needed

To really address the question raised by Yang and Zhang, we should do a second reduction to restrict the number of Wang tiles to only 5. This reduction from Wang tiles to a smaller set of 5 Wang tiles was not implemented.

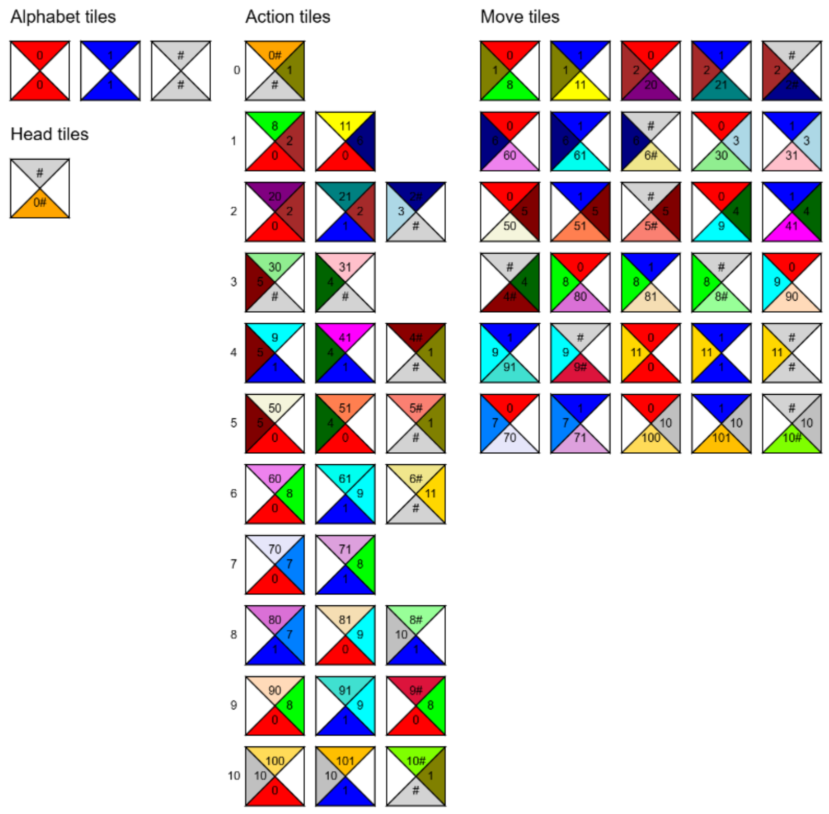


Figure 6: Tiles build with alphabet (0, 1, #) and our transition function

7 External help

The part concerning the Turing machine was partially made in collaboration with Emile Merian and Elliot Silberwasser who are working on ray tracing complexity.

The implementation of the reduction was inspired by two Git repositories [7][8].

References

[1] R. Berger. "The undecidability of the domino problem". In: *Memoirs of the American Mathematical Society* 66 (1966), pp. 1–72.

[2] S.W. Golomb. "Tiling with polyominoes". In: *Journal of Combinatorial Theory* 1.2 (1966), pp. 280–296.

[3] S.W. Golomb. “Tiling with sets of polyominoes”. In: *Journal of Combinatorial Theory* 9.1 (1970), pp. 60–71. ISSN: 0021-9800. URL: https://www.sciencedirect.com/science/article/pii/S0021980070800552.

[4] E. Jeandel and P. Vanier. “The Undecidability of the Domino Problem”. In: *Substitution and Tiling Dynamics: Introduction to Self-inducing Structures: CIRM Jean-Morlet Chair, Fall 2017*. Ed. by S. Akiyama and P. Arnoux. Cham: Springer International Publishing, 2020, pp. 293–357. ISBN: 978-3-030-57666-0. DOI: 10.1007/978-3-030-57666-0_6. URL: https://doi.org/10.1007/978-3-030-57666-0_6.

[5] Nicolas Ollinger. “Tiling the Plane with a Fixed Number of Polyominoes”. In: *Language and Automata Theory and Applications*. Ed. by Adrian Horia Dediu, Armand Mihai Ionescu, and Carlos Martín-Vide. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 638–647.

[6] *Pentominoes*. https://www.cs.brandeis.edu/~storer/JimPuzzles/ZPAGES/zzzPentominoes.html. (accessed 18.11.2024).

[7] N. Seriot. *Simulating Turing Machines with Wang tiles*. URL: https://seriot.ch/projects/simulating_turing_machines_with_wang_tiles.html.

[8] *Turing Machine Simulator written in JS*. URL: https://github.com/klimesf/turing-machine-js.

[9] C. Yang and Z. Zhang. *A proof of Ollinger’s conjecture: undecidability of tiling the plane with a set of* 8 *polyominoes*. 2024. DOI: https://doi.org/10.48550/arXiv.2403.13472. arXiv: 2403.13472 [math.CO]. URL: https://arxiv.org/abs/2403.13472.

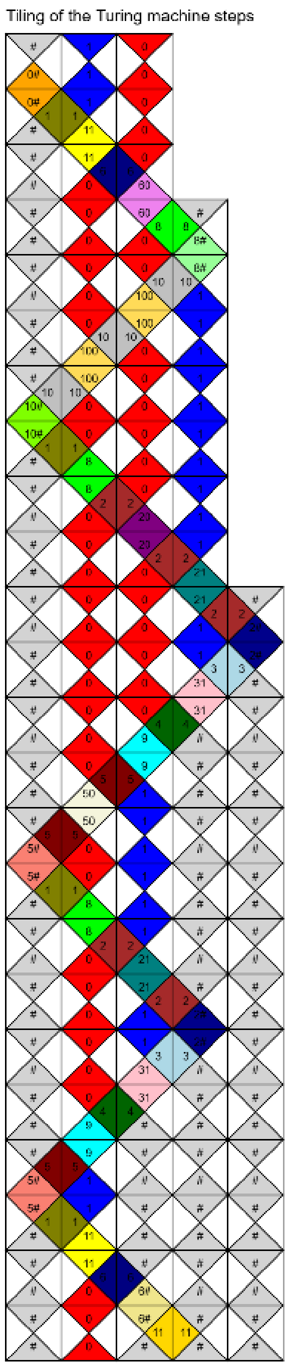


Figure 7: Reduction simulation for the word "#10"