
Electronic Thesis and Dissertation Repository

12-1-2017 11:00 AM

Self-Assembly of Tiles: Theoretical Models, the Power of Signals, and Local Computing

Amirhossein Simjour
The University of Western Ontario

Supervisor
Prof. Lila Kari
The University of Western Ontario

Graduate Program in Computer Science
A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy
© Amirhossein Simjour 2017

Follow this and additional works at: <https://ir.lib.uwo.ca/etd>



Part of the [Other Computer Sciences Commons](#), and the [Theory and Algorithms Commons](#)

Recommended Citation

Simjour, Amirhossein, "Self-Assembly of Tiles: Theoretical Models, the Power of Signals, and Local Computing" (2017). *Electronic Thesis and Dissertation Repository*. 5137.
<https://ir.lib.uwo.ca/etd/5137>

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlsadmin@uwo.ca.

Abstract

DNA-based self-assembly is an autonomous process whereby a disordered system of DNA sequences forms an organized structure or pattern as a consequence of Watson-Crick complementarity of DNA sequences, without external direction.

In this research we study the theoretical power of different mathematical models of self-assembly systems. The presented models are proposed to reduce the gap between self-assembly models and models that have been used in existing theoretical researches such as language theory and robotics. We propose self-assembly (SA) hypergraph automata as an automata-theoretic model for patterned self-assembly. We investigate the computational power of SA-hypergraph automata and show that for every recognizable picture language, there exists an SA-hypergraph automaton that accepts this language. Conversely, we prove that for any restricted SA-hypergraph automaton, there exists a Wang Tile System, a model for recognizable picture languages, that accepts the same language.

We also study complex self-assembly models and investigate the computational power of some variants of the Signal-passing Tile Assembly Model (STAM), as well as propose the concept of *Smart Tiles*, i.e., tiles with glues that can be activated or deactivated by signals, and which possess a limited amount of local computing capability. We demonstrate the potential of smart tiles to perform some robotic tasks such as replicating complex shapes.

Keywords: DNA based self-assembly, self-assembly of tiles, hypergraph automata, smart tiles

Co-Authorship Statement

This thesis consists of three research articles that are published or are submitted. For all the articles, the authors names are listed alphabetically. The contribution of each of the co-authors are listed below.

The article in chapter three "Hypergraph Automata: a Theoretical Model for Patterned Self-assembly." by Lila Kari, Steffen Kopecki, Amirhossein Simjour is published in International Journal of Foundations of Computer Science 25(4): 419-440 (2014).

LK- research idea, co-writing proofs of theorems;

SK- co-writing proofs of theorems, rewriting of the theorem 3.4.1, manuscript writing and editing;

AS- research idea, first manuscript draft, proof of the theorems.

The article in chapter four "Simplifying the Role of Signals in Tile Self-assembly." by Lila Kari, Amirhossein Simjour is submitted and is under review.

LK- research idea, co-writing proofs of theorems, manuscript writing and editing;

AS- research idea, first manuscript draft, proof of the theorems.

The article in chapter five "Smart Tile Self-Assembly and Replication." by Lila Kari, Amirhossein Simjour is published in Fundamenta Informaticae 154(1-4): 239-260 (2017) .

LK- research idea (specifically started the idea of definition of tiles with computational power), co-writing proofs of theorems, manuscript writing and editing;

AS- research idea, first manuscript draft, proof of the theorem.

Acknowledgements

I am glad to acknowledge all of those who played a role to complete this thesis, whether large or small.

I would like to take this opportunity to express how truly grateful I am to my supervisor, Dr. Lila Kari for her continuous commitment and support she has shown me throughout my study.

To my sister, Narges who supported me in Canada and played the role of my entire family, thank you.

I would like to thank my parents for all the support from thousands of miles away. Thank you for being there exactly when I needed you. Last but not the least, thanks to my wife Mahboubeh who have given me spirit and support in completing my thesis.

THANK YOU!

Contents

Abstract	i
Co-Authorship Statement	ii
Acknowledgements	iii
List of Figures	vii
List of Tables	x
List of Appendices	xi
1 Introduction	1
1.1 Hypergraph Automata: A Theoretical Model for Patterned Self-assembly	2
1.2 The Computational Power of Glue-deactivating Signals in Tile Self-assembly .	3
1.3 Smart Tiles and Replication	3
2 Literature review	5
2.1 Preliminaries	6
2.1.1 Mathematical Description	8
2.1.2 Computational Power	12
2.2 Error in Self-assembly	15
2.3 DNA-based Nanorobotics	18
2.4 2D Grammars	21
Bibliography	25

3	Hypergraph Automata: A Theoretical Model for Patterned Self-assembly	33
3.1	Introduction	33
3.2	Preliminaries	35
3.3	Hypergraph Automata	38
3.4	Hypergraph Automata for Picture Languages	45
3.5	Conclusion	56
	Bibliography	57
4	Simplifying the Role of Signals in Tile Self-assembly	60
4.1	Introduction	60
4.2	The Detachable Tile Assembly Model	62
4.2.1	Formal Definition of DTAM	64
4.2.2	Transitions	66
4.3	Turing Universality	72
4.4	Self-assembly of Thin Rectangles	79
4.4.1	Informal description of the DTAM tile assembly system	79
4.4.2	Detailed description of the DTAM tile assembly system	82
4.4.3	DTAM tile complexity and waste analysis	95
4.5	Conclusions	96
	Bibliography	96
4.6	Addendum	99
5	Smart Tile Self-Assembly and Replication	101
5.1	Introduction	101
5.2	Smart Tile Assembly	103
5.2.1	Basic Definitions	103
5.2.2	The Smart Tile Assembly Model	106
5.3	A Smart Tile Assembly System that Replicates L -Convex Shapes	113
5.4	Discussion and Future Work	126

Bibliography	127
6 Conclusion	130
A Appendix: Examples of SA-Hypergraph Automata	132
Curriculum Vitae	137

List of Figures

2.1	DAO and DAE tiles	8
2.2	An example of a tile assembly system with a seed	11
2.3	An example of flipped tiles	17
3.1	An example of Wang tile system	37
3.2	The self-assembly of a single coloured pattern	38
3.3	Transition in hypergraph automata	41
3.4	Underlying graph of the SA-hypergraph automaton that constructs a coloured pattern	42
3.5	Step by step construction of a picture graph	44
3.6	Hyperedges of a hypergraph automaton A that recognizes a recognizable pic- ture language L	47
3.7	Want tile system that accepts the same language as an SA-hypergraph automaton	51
4.1	A detachable tile in the DTAM	64
4.2	An example of a general configuration and the assembly-graph associated to it .	68
4.3	An example of an attachment transition	70
4.4	Simulation of a Turing machine using a deterministic zig-zag tile assembly system at temperature 2	74
4.5	The original tiles from the deterministic zig-zag tile assembly model at temper- ature 2, and their corresponding gadgets made out of SDTAM tiles	75
4.6	The tiles that are used in the bit reader part of the gadgets	76
4.7	The idea of the steps that replace an $M \times (M! + 2C)$ rectangle by an $(M + 1) \times$ $((M + 1)! + 2C)$ rectangle	80
4.8	The use of the mirror rectangle in preventing unexpected attachments	81

4.9	The attachment of two rectangles and start of the column replacement	83
4.10	A thin rectangle that is the input of Step 1 of the construction, and the glues on the tiles of its bottom edge	84
4.11	The tile types that are used in the construction of a thin rectangle and their respective glues	85
4.12	A detailed view of Step 2, and the attachment of the first column in Step 3 of the construction of a thin rectangle	87
4.13	Step 3 (expansion), and Step 4 (reattachment) of the construction of a thin rectangle. Part 1	88
4.14	Step 3 (expansion), and Step 4 (reattachment) of the construction of a thin rectangle. Part 2	89
4.15	The tiles that are involved in the attachment of the left and right parts of the rectangles	95
5.1	An example of an attachment transition	108
5.2	A smart-TAM system at temperature 1 with a single tile type that can assemble a final configuration consisting of a row of three tiles	112
5.3	An example of an L -convex supertile	114
5.4	An example of an L -convex supertile with the border glues needed for its replication by the smart-TAM system	116
5.5	The general idea of the replication process of an L -convex supertile of a given shape, by a smart tile self-assembly system	117
5.6	The bijective mapping between the glues on the border of the original supertile and the glues on the border of the quarter-rectangle	117
5.7	The result of Step 1: Filling all four quarter-rectangles around the original supertile	118
5.8	The rectangular hole that is obtained at the end of Step 2	119
5.9	The tile set $\Theta_{border-ne} = \{T_1, \dots, T_6\}$, and $\Theta_{replica} = \{T_{replica}\}$	120
5.10	The local tile computational device (GSM)	121

5.11	High-level design of another smart tile assembly system that replicates arbitrary shapes	126
A.1	Example 1 - An example of a language of coloured self-assembled patterns . .	133
A.2	Example 2 - An example of a language of coloured self-assembled patterns . .	134
A.3	Example 3 - An example of a language of coloured self-assembled patterns . .	135

List of Tables

4.1	The list the signals and transitions for all the tiles types, except the dark grey border areas. Part 1	93
4.2	The list the signals and transitions for all the tiles types, except the dark grey border areas. Part 2	94

List of Appendices

Appendix A Examples of SA-Hypergraph Automata	132
---	-----

Chapter 1

Introduction

Using natural systems as a tool for computation is one of the methods of unconventional computation. In this thesis, we focus on formal models of natural computation that uses DNA as their primitive tool for computation. Our research is mainly focused on self-assembly process. DNA-based self-assembly is an autonomous process through which nanoscale DNA-based structures join together to build a larger structure without external control. The self-assembly model has been used for both computation and for building nanostructures. Self-assembly is not limited to DNA-based systems. Atoms, for example, react to each other to form molecules, molecules interact with each other to build crystals, etc. The common feature of all these processes is that small building blocks self-assemble to form a structure that reduces their free energy level. In this chapter, we will review the self-assembly of DNA structures, which is used for building nanostructures and nanorobotics and also can be used for computation.

The DNA-based self-assembly system starts with single stranded DNA sequences. Using these DNA sequences, some macro-molecular building blocks which are called *tiles* are built. Each tile has 4 sticky ends, called *glues*, to connect to other tiles and interact with them. Each of these sticky ends is a single stranded DNA sequence that can attach to its reverse complement single stranded DNA sequence. Attachment of tiles builds a larger structure called DNA lattice. For the remaining of this article, this system will be referred as *Tile Assembly-System* or TAS.

Tile assembly systems are autonomous, as a result, the control over the system is limited. The limited control leads to some advantages and some disadvantages. Large number of processors can be used simultaneously with minimum energy cost, but this limited control will

make the design more complicated. In the next chapter, an introduction to the self-assembly systems and related topics is presented. The next three chapters after the introduction consist of three articles, each trying to enrich our understanding of self-assembly with DNA tiles by defining and studying theoretical models of self-assembly.

The SA-hypergraph automaton introduced in Chapter 3 is an attempt to make a connection between language theory and the theory of DNA-based self-assembly systems: We addressed the relationship between recognizable picture languages and automata that are capable of simulating a particular type of self-assembly systems in which a pattern is constructed. In Chapter 4, we continued our research on theoretical models of DNA-based self-assembly systems with investigating the power of tile assembly system with limited use of signals which can only deactivate connections between tiles. Lastly, in Chapter 5, we present a new model for complex tiles, called *smart tiles*, that are enhanced with a computational device as well as the ability to have limited communication with their neighboring tiles.

In the following we present a brief summary of Chapters 3, 4, and 5.

1.1 Hypergraph Automata: A Theoretical Model for Patterned Self-assembly

Patterned self-assembly is a process whereby coloured tiles self-assemble to build a rectangular coloured patterns. We propose self-assembly (SA) hypergraph automata as an automata-theoretic model for patterned self-assembly. We investigate the computational power of SA-hypergraph automata and show that for every recognizable picture language, there exists an SA-hypergraph automaton that accepts this language. Conversely, we prove that for any restricted SA-hypergraph automaton, there exists a Wang Tile System, a model for recognizable picture languages, that accepts the same language. The advantage of SA-hypergraph automata over Wang automata, acceptors for the class of recognizable picture languages, is that they do not rely on an a priori defined scanning strategy. This research has been completed and published as *Hypergraph Automata: A Theoretical Model for Patterned Self-assembly* in *International Journal of Foundations of Computer Science (IJFCS)*, volume 25, 2014, pages.

1.2 The Computational Power of Glue-deactivating Signals in Tile Self-assembly

Sending signals through DNA-based structures is a method to control the DNA-based self-assembly systems. Signals are implemented using DNA strand displacement and make it possible for tiles to change the current state of other tiles that are attached to them. Although currently there is no experimental result that demonstrates building tiles with glue deactivation signals, the theoretical design that we propose is plausible. There are some previous researches to understand the power of signals, however the power of detachment signals in self-assembly systems is not well-studied yet.

In our research we show that the detachment signals make DNA-based self-assembly systems Turing universal at temperature one. Moreover, we prove that the construction of thin rectangles only using detachment signals needs fewer number of tile types.

The article that is presented in Chapter 4 is under review.

1.3 Smart Tiles and Replication

We investigate the computational power a variation of the signal-passing tile assembly model (STAM), as well as propose the concept of *smart tiles*, i. e., tiles with glues that can be activated or deactivated by signals, and which possess a limited amount of local computing capability. This is a formal model, independent from implementation details, but there is no reason to believe that small computing devices attached to tiles could not be feasible experimentally. We demonstrate the potential of smart tiles to perform robotic tasks such as replicating complex shapes.

The significance of this research is the definitions of three new models for self-assembly systems. Each of these models are proved to have applications and improve the existing models of self-assembly systems.

It is hoped that this research will empower the scientist with a set of new models for self-assembly systems. Moreover, with introducing these new models it is hoped to connect the theory of DNA-based self-assembly system to potential new areas such as robotic systems. The

connection between these fields makes it possible to apply theoretical results from DNA-based self-assembly systems to robotic systems where robots have limited computational power.

Chapter 2

Literature review

The experimental DNA-based assembly system was introduced by Seeman in 1982 where he used DNA strands to build DNA-based junctions [62]. Winfree et al [74, 70] then used self-assembly systems to build DNA-based lattices. Winfree in his Ph.D. thesis [71] introduced 2D DNA-based self-assembly systems, and he proved that the 2D self-assembly can simulate a Turing machine. Adleman wrote a mathematical description [3] of assembly systems in one-dimensional space based on “step counting” and probabilistic distribution of events during the time, but computation power of 1D self-assembly was very limited.

Winfree’s model of DNA-based self-assembly is based on Wang tiles [71]. A Wang tile is a square with a colour on each side and tiles with matching colours attach to each other to form structures. Wang tiles have also been used for the classical domino problem [69] and had some applications in image processing [22]. Winfree’s mathematical model for self-assembly has been modified to solve wide range of mathematical problems [5][40][44]. Winfree in [59] introduced one new complexity measure based on the tile types and Adleman in [4] described a new complexity measure based on the construction time. Kari in [4] and [44] introduced mathematical models based on new design for tiles, and Doty and Kao in [29][41] changed the environment description and mentioned a new model. Kao and Schweller defined the equivalence of approximation algorithms for building structures [42] and Doty introduced a model based on randomized algorithms to build exact shapes [27]. Moreover, Winfree [70], Kari[16] and Summers [30] studied the computational power of self-assembly systems.

Fault reduction is one of the most important parameters in the design and implementation of

a self-assembly system. In the earliest methods of development, error rate was approximately 10 percent per each tile [60]. That error rate was significant and a model with 10 percent of error was not suitable neither for computation nor building patterns. Chen [19][18], Winfree [73], Doty [29], Reif [55] and Sahu [60], have reduced the error.

We can divide error and fault reduction methods into two different categories: methods which change the structure or design of tiles, and methods that change the environment or change the method of the design of the systems. In the first category we can see algorithms which used larger tile or used more than one tile for each position. For the second category we can mention Winfree's paper [9], which proposed utilizing a seed as a starting point and reduced the error rate with this method. Moreover, a number of methods based on modifications on the temperature [29] or change of the growth method [73] will be discussed.

In Section 2.1, the basics of self-assembly from mathematical points of view is described, then the probable problems such as complexity and design of new algorithms in this domain, are mentioned. The focus of the Section 2.2 is on the error reduction and description of different models for making a self-assembly system fault-tolerant. Although error reduction is not the main focus of this thesis, some of the methods that are used in fault reduction, such as replacing a tile with a gadget or using shape restrictions, are very similar to the methods that we have used in our proof and constructions in Chapter 4. In Section 2.3 the nanorobotic and its relation to DNA computation is explained. Section 2.3 is devoted to a review on 2D languages and their potential relation to self-assembly systems.

2.1 Preliminaries

Seeman in 1982 [62] used DNA sequences to self-assemble a DNA-based junction. Afterwards, Adleman [1] in 1994 used one-dimensional DNA-based self-assembly as a computational tool. Adleman used single-stranded DNAs for computation. It is notable that Adleman's DNA-based system solved an instance of an NP-complete problem. Due to the fact that the number of DNA strands that were used in his design was growing exponentially in the input size, Adleman's solution was not exponentially scalable. However, his work was the first instance of using DNA strands for computation. Winfree [71] used DNA-based structures called

tiles with four sticky ends on the corners and introduced the *abstract Tile Assembly Model* [71], as the mathematical model to describe the DNA-based tile assembly systems. The common feature of all self-assembly processes is that small building blocks self-assemble to form a structure that reduces their free energy level [52].

Winfree used 1×1 squares with colours on the edges as an abstraction for a DNA tile. Moreover, he defined a mathematical model for the self-assembly of tiles. Winfree used self-assembly for both building nanostructures and performing computational tasks, and proved that self-assembly in 2D can simulate a Turing machine [71].

Winfree used Wang tiles for the mathematical description of the tiles. Here we briefly review some of the results on construction of DNA-based tiles. [46] shows different methods of tile constructions with DNAs. After Winfree, others used different designs for tiles [46]. Some of these structures have more connection points and some other use different angles between connection points. Yan successfully marked some of the tiles by gold particles, Mg⁺ ion, and some other materials [39]. Moreover, Winfree in [23] suggested that by applying such embedded data to each tile, classical circuits can be built in small spaces. Yurke et al. [17] successfully added some quantum dots to the tiles and proposed the usage of self-assembly as a platform for the Quantum Dot Cellular Automata [11].

There are several different implementations of Wang tiles with DNA. It is notable that all these tiles that we review here have single stranded DNAs at their corners. Figure 2.1 shows the DAO and DAE tiles, which were introduced by Winfree [74, 71]. Here D stands for double, A stands for antiparallel, and O or E denote the odd or even number of half turns between crossovers.

In mathematical models for tile assembly systems, all these tiles are represented by 1×1 squares with colours on the edges. The direction of sequences, and the number of half-turns are not included in abstract mathematical models of self-assembly systems.

In the process of building a flat structure, DAO tiles are preferred to DAE tiles. The DAE tiles have an extra “twist” in their structure that results in their assemblies curving out of the plane and forming tubes [57]. The DAE tiles can be used to build 3D structures. Another concern about the DAO tiles is that, in general, each tile type in the Tile Assembly Model (TAM) requires two versions in the DNA implementation; one version with both 5’ sticky

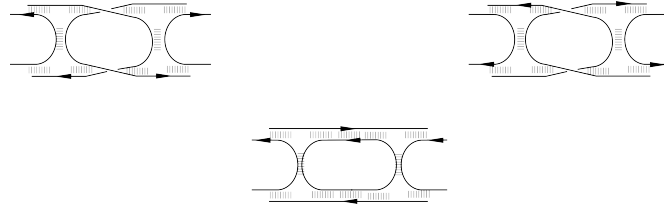


Figure 2.1: Two implementations of DAO tiles on the first row [74, 71]. One of the implementations of DAE tiles [74, 71] on the second row.

ends positioned on top, and the other one with both 5' sticky ends positioned on the bottom. In some cases [8][58], a tile system only uses a tile in a subset of positions, as a result the implementation of the both version of the tiles might not be necessary.

Design of the tile types is not limited to DAO and DAE tiles. [45] designed TAE and TAO tiles, where T means that both designs have three double-helices linked by strand exchange. [45] also designed a new process to build a tile from single strands.

Many tile implementations are not similar to DAE or DAO tiles. For example, [39] designed a tile with the ability to attach to up to four other tiles, but this tile uses more than one single stranded DNA on each corner as the sticky end.

2.1.1 Mathematical Description

The first mathematical model for self-assembly systems uses Wang tiles for describing the DNA-based tiles. A Wang tile is a 4-way domino. In other words, a Wang tile is a unit square with a colour on each edge. A function from the two-dimensional space \mathbb{Z}^2 to a set of Wang tiles Θ is called a *tiling* over Θ . A tiling is called a *valid tiling* if and only if all common edges of any pair of adjacent tiles have the same colour. Sometimes we replace the colours with numbers. Wang [69] in 1961 defined the *plane tiling problem* as following:

Definition : Plane tiling problem ([69]). Given a set of Wang tiles determine if this set can build a valid tiling of the plane or not.

Wang conjectured that a set of 4-way dominoes can tile the entire infinite 2 dimensional area if and only if they can tile that area periodically [69]. Berger in 1966 found an aperiodic tile set which could cover the entire plane [10]. Moreover, Berger proved that the plane tiling

problem is undecidable [10]. He used a tile system to encode the Halting Problem into the plane tiling problem. For a long time finding the smallest aperiodic tile set that can tile the entire 2 dimensional space was an interesting problem for mathematicians. Berger's set consist of 20426 tile types, but currently smallest known set has only 13 different tile types [24]. Culik [24] in 1996 found an aperiodic tile set with 13 tiles. He followed Kari's [43] approach to the plane tiling problem. Kari's method was based on sequential machines that multiply Beatty sequences of real numbers by rational constants, using this method Kari had found an aperiodic tile set which consist only 14 tiles.

Another famous problem in this area is the Periodic Tiling Problem.

Definition Periodic tiling problem([38]). Given a set of Wang tiles determine if this set can build a periodic tiling or not.

Koriakov in 1972 studied this problem [38], and he proved that the periodic tiling problem is undecidable. The plane tiling problem for infinite space is interesting and well-studied, however finite areas are also interesting from practical point of view. In this topic one of the interesting problems is to determine whether a specific shape can be covered with a set of predetermined domino tiles or not.

There are two important differences between domino problems and self-assembly systems. First, in domino problems usually a predefined set of tiles exists. Second and more importantly, in domino problems finding a final shape is the goal and the steps are not important.

The abstract Tile Assembly Model was originally proposed by Winfree [71]. The aTAM extends the theory of Wang tiles [69] and includes a mechanism for tiles to stick to each other and grow into a structure. Just like tiles in domino problem, each Wang tile in aTAM is a square with one colour (or number or letter) called glue on each edge. It is notable that in aTAM tiles can not be flipped or rotated. Formally, a tile is a unit square with edges labelled north, east, south and west, and one glue on each edge. For a tile t , the glues on its four edges are denoted by $\sigma_N(t)$, $\sigma_E(t)$, $\sigma_S(t)$ and $\sigma_W(t)$.

Informally, we can assume that a tile can move freely in a 2-dimensional environment and when it collides with another tile they will stick together if their glues match.

Formally, a tile assembly system in the aTAM is a triple $\langle T, g, \tau \rangle$ where T is a finite set of

tiles, $\tau \in \mathbb{Z}_{>0}$ is the *temperature*, and g is the *glue strength function* from $\Sigma \times \Sigma$ to N where Σ is the set of edge labels and N is the set of natural numbers. It is assumed that $null \in \Sigma$ and $g(x, y) = g(y, x)$ for $x, y \in \Sigma$, and $g(null, x) = 0$ for all $x \in \Sigma$.

A *configuration* is a map from \mathbb{Z}^2 to $T \cup \{empty\}$. For $t \in T$, $\Lambda^{x,y}_t$ is the configuration such that $\Lambda^{x,y}_t(i, j) = t$ if and only if $(i, j) = (x, y)$ and empty otherwise. Let C and D be two configurations. Suppose there exist some $i \in T$ and $(x, y) \in \mathbb{Z}^2$ such that $C(x, y) = empty$, $D = C$ except at (x, y) , $D(x, y) = i$, and $g(\sigma_E(i), \sigma_W(D(x+1, y))) + g(\sigma_W(i), \sigma_E(D(x-1, y))) + g(\sigma_N(i), \sigma_S(D(x, y+1))) + g(\sigma_S(i), \sigma_N(D(x, y-1))) \geq \tau$. Then we say that the position (x, y) in C is *attachable*, and we write $C \rightarrow_T D$ to denote the transition from C to D in attaching tile i to C at position (x, y) . Informally, an attachment transition denoted by $C \rightarrow_T D$ means that D can be obtained from C by adding a tile to it such that the total strength of the interaction in adding the tile to C is at least τ . The self-assembly proceeds by a succession of attachments of tiles to an existing structure.

For simplicity it is assumed that the attachment transitions in a self-assembly systems start from a configuration that only has one specific tile, which is called *seed*. Practically the system can be controlled in a way that with a high probability transitions start from the seed. Sometimes, a *supertile* (a structure consisting of several tiles attached to each others) is used as the starting point of the transitions. A supertile can be used as the starting point of the transitions of an assembly system or can be the result of adding some tiles to the seed. A tile system can be defined as a quadruple $\mathbb{T} = \langle T, S, g, \tau \rangle$, where S is a supertile called seed supertile. Figure 2.2 part i) shows an example of a tile assembly system with 3 tiles. The tile with grey background is the seed. Let assume that the temperature is one and all the glues have strength 1. We start with a configuration that consists of one tile in part ii). Part iii) shows the result of applying an attachment transition to the configuration in part ii). Part iv) shows the result of the attachment of the last tile to the configuration. No further attachment transition is possible after part iv).

Intuitively, if the tile system \mathbb{T} produces only one possible final super tile A , we say that the tile system uniquely produces A .

The models of tile assembly systems are not restricted to this mathematical description. Kari in [44] proposed a different model based on triangular tiles, and in [28] studied another

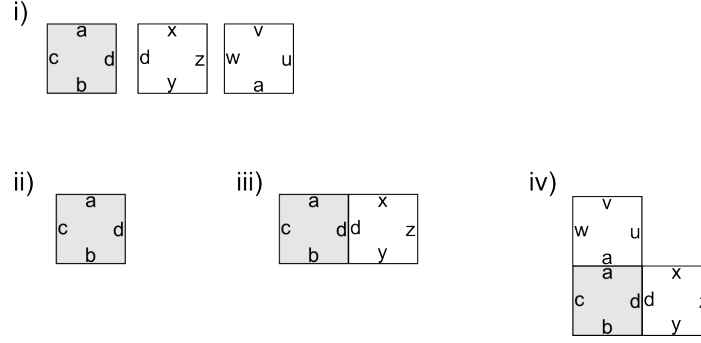


Figure 2.2: Part i) shows three tiles that are used in this tile assembly system. The temperature of the system is $\tau = 1$, and the strength of all glues is equal to one. The tile that has grey background is the seed tile. Part ii) shows the initial configuration with one seed tile. Part iii) shows the attachment of the second tile to the seed tile, and part iv) shows the final configuration. No other attachment is possible at this point.

model based on the negative glues. Kao et al. in [5] described different models and in each of them considered some of the properties of the process which the basic model Winfree ignored for the sake of simplicity.

Kari discusses the DNA-based self-assembly systems based on triangular tiles instead of square tiles. They were considered as different models of triangular tiles, particular right triangular tiles and equilateral triangular tiles. Both of these models have been physically implemented based on DNA sequences. However, even without implementation, proposing new models is still useful and new models can result to finding new applications and different implementations. Kari showed that the triangular tile assembly systems are as powerful as the Turing machine.

Kari also considered negative weight of glue function and proved that using negative weight, we can simulate computation with smaller size of the final shape [28]. It is an important result because the size of the final shape has some effects on the error rate and speed.

Jonoska and McColm used flexible tiles [40]. These tiles do not have firm sticky ends and their molecules can change positions of their sticky ends. This method will convert the final assembly shape to a graph. [40] explained the flexible tile's theory.

Kao et al. [5] compared several modifications on TAS.

The Multiple Temperature Model. In the multiple temperature model, it was suggested to change the temperature during the time, and in each step it was assumed that the previous

step is completed. It means that when step i was started, all the possible interactions between tiles in the step $i-1$ were done. Kao referred to the system with k different temperatures to be a k -temperature system. Using this method, the process will be more controllable, and as a result some shapes are constructed with fewer number of tile types.

The Flexible Glue Model. In the flexible glue model, a partial match between two DNA sequences has been considered. Consequently, the restriction of $g(x,y) = 0$ for $x \neq y$ is eliminated. Using this method, the size of the minimum tile set to build some of the shapes will decrease. However, practically all the arbitrary strength functions cannot be designed.

The Unique Shape Model. In the unique shape model, Kao redefined what we mean for a system to uniquely produce a shape S . In this model, a tile system uniquely produces a shape S if the only terminal supertiles produced by the tile system are of shape S . Thus we allow the system to produce many different supertiles as long as they all have the desired shape.

The q-Tile Model. In the q-tile, it is assumed that tiles can build supertiles and supertiles can attach to each other. The size of the supertiles is considered to be limited.

These methods can reduce both assembly time and minimum number of tile types.

2.1.2 Computational Power

Computational power of aTAM has been studied by Winfree [71][70]. Winfree proved that aTAM is as powerful as Turing machine. In other words, any Turing machine can be simulated with the aTAM. The proof is based on Bounded Cellular Automata (BCA). It is known that BCA is computationally universal. Winfree proved that all the rules of any BCA can be encoded in the tiles of an aTAM. Winfree used the deterministic aTAM and temperature two for universal computation.

In addition to the deterministic model, non-deterministic TAM has an important role in problem solving [15]. It is trivial that the non-deterministic form is at least as powerful as the deterministic form. Moreover, it is proved that for the infinite shape construction, deterministic TAM is strictly included in non-deterministic TAM [16]. Moreover, in some cases non-determinism can reduce the program size complexity [16]. It is noticeable that, although it is possible to reduce the tile complexity using a non-deterministic model, it is harder to find the

minimum tile type set needed to build a structure non-deterministically. Solving the minimum tile set problem in a deterministic form is a NP-Complete problem, however, this problem is more complicated for nondeterministic TAM, and it is Σ_2^P -complete.

The power of aTAM depends on the temperature of the operation [30]. [30] proved that using negative glues can increase the power of self-assembly models. The TAM of temperature one with negative glues is as strong as the aTAM at temperature two, and it can make the general-purpose computation possible. Moreover, [54] proves that only one single negative glue is enough, and temperature one in addition to one negative glue can simulate Turing machine and general-purpose computation.

There exist two important measures to compute the complexity of the self-assembly algorithms: time complexity and program size complexity. Here, these two measures are briefly described and some examples are given.

The set of the tiles which stick together and built the final configuration can be considered as a program. Winfree et al. [66] proved that the minimum set of tile types that can build a specific shape and the shape's descriptonal complexity are related. They proved that the minimal set of tiles to form a shape can be bounded both above and below in terms of the shape's Kolmogorov complexity.

In addition to the tile complexity, Adleman et al. [4] introduced a new measure based on the time. Informally, the time complexity of a shape is the minimum time that is needed to build that shape using maximum parallelism.

Based on these two complexity measures, self-assembly algorithms can be compared. For example, there are various kinds of self-assembly systems that can build a square. Finding an efficient way to construct a square was one of the earliest problems on the self-assembly systems. Rothmund and Winfree [66] constructed a square with $\Theta(\log(n))$ tile types and in minimum time of $\Theta(n \log(n))$. Afterwards, Adleman [4] used a similar idea to construct a square in time of $\Theta(n)$ and the optimal program size $\Theta(\frac{\log n}{\log \log n})$. According to Kolmogorov complexity, both the time and tile complexity of these algorithms are optimal.

It is noticeable that these lower bounds are computed based on the aTAM. In other models, it is possible to reach better solutions. For example, using flexible glues, a thick rectangle can be built with only $\sqrt{\log n}$ tiles, and using the q -tile model a thin rectangle can be built with only

$\frac{\log n}{\log \log n}$ tiles [5].

Moreover, Kari [16] demonstrated that in some of the problems, using nondeterminism, the program-sized complexity can be reduced by a linear factor.

Self-assembly systems can be used as a tool for computation, from simple mathematics operation to solving instances of NP-complete problems, and simulating Turing machine. The first implementation of tile assembly systems (TAS) started with construction of a counter [8] and other primitive mathematical operations. Despite the fact that counting is a simple operation, the implementation of this counter practically shows TAS's power.

After that simple operations, many other algorithms on constructing simple shapes have been proposed. Winfree [66] proposed an algorithm to construct any square with arbitrary width. This algorithm was based on using a counter to build a thin rectangle and using a constant number of tiles to convert that rectangle to a square. He used $\Theta(\log n)$ tile types and assembled the square in $\Theta(n \log n)$ time. Adleman [4] used the same idea but with a smaller tile set. He used a different counter which was able to work in parallel. Moreover he changed the base of counting to reduce the number of tiles. As the result, he reached the time complexity of $\Theta(\log n)$ and program-size complexity of $\Theta(\frac{\log n}{\log \log n})$.

Adleman [2] proved that the minimum number of tile types to build any arbitrary tree can be computed in polynomial time. He used the similarity between subtrees, and used the same tile for similar positions in the tree. Based on this experiment, any tree can be constructed with this optimum solution in $\tau = 1$.

Self-assembly can be used to implement complex algorithms. Considering its computational power, it is clear that any algorithm can be implemented using a self-assembly system. There are several solutions for NP-complete algorithms based on self-assembly systems. These algorithms usually have exponential times and use non-constant program-sizes. It is notable that these theoretical solutions are relaying on the assumption that unlimited copies of tiles are available, which is not practically true. However, they also show the potential power of self-assembly systems to deal with complex problems. Recently, Brun [15] proposed a new tile system with $O(1)$ tile types which decides 3-SAT by $O(1.8393^n)$ assemblies in parallel. This algorithm also can be executed in $O(n)$ time non-deterministically. Using a recursive program flow is the most important point in this new algorithm. The proposed 3-SAT algorithm can be

easily generalized and be used for other studies. The main idea is to build a random solution in a line, then for each step to assign some value to the last clause, apply this value to all of the previous clauses, and continue the next step by a smaller line.

In addition to mathematics systems, algorithms to construct shapes, and algorithms for decision problems, there are some other kinds of theoretical concepts in self-assembly algorithms. Using the nature of self-assembly algorithms, different approaches for approximation algorithms [42] and probabilistic algorithms [27] are proposed. Approximation algorithms in this context mean building a shape close to the target shape. Probabilistic algorithm means designing a TAS which forms the desired shape with high probability.

2.2 Error in Self-assembly

A major challenge in the practical DNA-based assembly systems is the design and the implementation of fault tolerant systems. Faults can affect computational tasks and pattern construction. Prior designs of the self-assembly systems had error rates between 0.5% to 5% or even more [8]. It is obvious that such error rate is deleterious to the construction of large scale structures. Elimination of the errors is one of the most challenging concepts in self-assembly systems. In some cases, it is possible to reduce this error through heuristic designs [8]. This means, a specific design of a tile assembly system might reduce the error rate significantly. Although special designs might be useful, they are not general. Moreover, finding some new methods to reduce the error rate to an acceptable range can be useful. Several different methods to control the error rate are explained as follows.

It is possible to affect the assembly error rate by changing the tile design. For example, it is notable that the strength of attachment of mismatched glues is in practice not zero. Indeed, the “sticky ends” can be partially complementary which means that they can still attach, albeit by a weaker connection. In order to reduce these unexpected attachments, we can change the physical parameters of the experiments to make the expected attachments stronger and minimize the strength of unexpected attachments. One way to achieve that is to use longer single-stranded DNA as sticky ends, and another is to use more G/C in the DNA sticky ends, to make their attachment stronger, and a third way is to decrease the temperature of the experimental system.

Note that the third method comes at the expense of slowing down the assembly process.

Using non-standard models of the self-assembly system also has some effects on the error rate. As an instance, putting some limitations on the maximum parallelism can reduce the error rate as well [56].

Finally, using redundant data can result in a smaller error rate. For example it is possible to design a fault-tolerant algorithm through data to the third dimension of the space [19]. [56] used another method to reduce the error. They used dependency between tiles and designed a tile set in which the occurrence of error in one position immediately leads to another error in one of its neighbours. Using this method, they can reduce the error to ε^2 or less; where ε is the probability of the appearance of an error in each tile.

In the remaining of this section, first, a review on the fault-resistant systems based on the changes in the environment is given. Then, the methods based on the changes in designs are reviewed.

Changing some of the properties of the self-assembly systems' environment can lead to a system that is more resistant against the fault. Winfree [72] explained that the lowest error rates occur at the melting temperature. It must be noted that, at the melting temperature, the growth rate of the crystal has its smallest value. Low growth speed is not practically useful, therefore the lowest error rate cannot be achieved in practice. In order to achieve a fault-tolerant system, it is also possible to change the concentration and the strength of the sticky connections. In this case, the error can be reduced to an arbitrary rate close to zero.

Using a seed and changing of the design based on seed can also reduce the error [61]. In theory, the seed is a tile that the self-assembly process always starts with. Practically, the design of the seed and using the seed can reduce the number of errors, and there is an arbitrarily large kinetic barrier to unseeded growth [61]. Moreover, the design of a larger seed can reduce the error and can increase the speed of the assembly process. For example, using the DNA origami to construct a large seed [9] can reduce the program-size complexity and reduce the error rate. Winfree [9] used a DNA origami seed with 32 starting points to build a counter.

In addition to environmental variables, a number of changes in the design of a self-assembly system can make it more resistant against errors. Some of these changes result in new models and some of them can be described with the standard model. Chen and Goel [18] used a snaked

proof-reading for error-correction scheme and proved that replacing each tile with $k \times k$ block of tiles can reduce the error. The most important point about their method is that they did not change anything in aTAM. Doty [29] used a fuzzy temperature model. In the fuzzy temperature model, temperature two has been normally used but sometimes the temperature can change to one and allow some of the tiles to stick to the supertile with strength one. In this model, the structure will be more stable with the change of the temperature. The idea is to avoid the usual lab errors with a different design of the algorithms. Doty used this model to construct a square based on aTAM.

Yin et al. in [55] used a different model which resulted in the same output. Although previous methods built a larger structure for each tile system, Yin described a model in which the error has been reduced without increasing the size. They forced the structure to have a stronger dependency on the neighbours. In this structure, the occurrence of each error immediately results in the occurrence of another error. Yin et al. proved that this dependency can reduce the error to ε^2 or ε^3 , where ε is the probability of an incorrect attachment.

Using a three dimensional space can result in an error resistant self-assembly system [19]. Using the 3rd dimension will increase the number of tiling assemblies less than previous models, and decrease the error as much as previous methods. It is noticeable that using the third dimension will not change the scale of the pattern in the plane. Redundancy will be added only to the 3rd dimension.

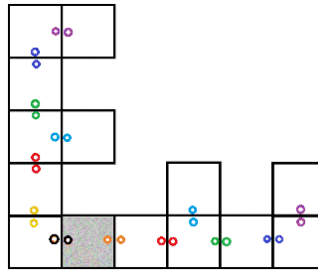


Figure 2.3: An example of using flipped tiles. The grey tile is the *seed*. Since the left subtree is the flipped version of the right one, the same tile set can be used for both of them.

In standard TAM, it is assumed that tiles cannot rotate. In the related problems in tiling, always the same assumption exists [21][51]. Rotation problems usually ignore the transformation of the tiles. For example, Gales and Rapaport [35][36] studied the rotation board. A

rotation board is a $N \times N$ square of tiles which colour mismatches. The problem is: Given a rotation board as input, is it possible to convert the board to a valid tiling of the square if rotation of tiles is the only allowed operation? Gales and Rapaport proved that the rotation board problem is NP-complete. Moreover, they studied the optimization version of the rotation board problem. The goal in the optimization version is to find a lattice with minimum mismatches. This problem also is a NP-complete problem.

[51] and [75] defined a specific version of the board problem. These new versions allows both rotation and local moves. However, in this version, colours of the borders must be predefined.

It is noticeable that, since a tile can attach to a rotated version of itself, tiling problems in general cannot allow rotation. However, in experimental self-assembly systems tiles cannot attach to their rotation version (See Figure 2.3). The complementary properties and 3' and 5' ends prevent the attachment of a tile to itself. In a few implementations of self-assembly tiles, rotation is allowed [39]. This kind of implementation results in infinite structures. Moreover, some of the simulation tools can simulate rotation and flip [50][67]. The standard implementations of the Wang tiles cannot rotate. However, DAE tiles and one of implementations of the DAO tiles can flip. The only allowed flip in these tiles is one diagonal flip. Using the flipped tiles reduces the program size complexity. For example, Figure 2.3 shows a tree structure and its tiling. Without the flipped tiles the construction of the tree structure of Figure 2.3 needs five more tiles.

Adleman et al. [2] proved that the minimum tile set problem is NP-complete. With a few changes in the solution of the minimum tile set problem, it is clear that adding flip ability to tiles does not change the complexity. Flipped tiles can reduce the program size complexity by two times, and they will not increase the complexity of finding minimum tiles.

2.3 DNA-based Nanorobotics

Chapter 5 introduces the smart tile assembly model, which is based on tiles that are each enhanced by a computational device. The smart tile assembly model is then used to perform tasks similar to robotic systems [34]. This section gives a brief description of DNA-based

nanorobotics where we can see the power of DNA based systems to build complex nanorobots and computational devices. Moreover, it is notable that similar ideas that are used for building these nanorobots can be used to build complex communicating systems between tiles [53].

DNA-based walkers move on a set of predefined platforms, and DNA-based self-assembly systems have an important role in the design of these platforms. The construction of the platforms can be considered as a pattern problem. This section focuses on the following two aspects:

- **Non-autonomous nanorobotics.** This part reviews the design and implementation of nanorobots which are sensitive to the environment, whereby using the environmental variables or adding/removing other materials, it is possible to give some instructions to the robot.
- **Autonomous nanorobotics.** The second part of this section explains the design and implementation of nanorobots which can act without any external control.

Using biological material and specially DNA strands was an important step in towards the goal of building nanorobotics. These devices can act on a small spaces. Moreover, using DNA sequences results in simpler designs. For example, Yan [32] reported the construction of a DNA based device, which can change its state in different situations. Using DNA can result in reversible systems and Yan used this property to change between two different states. This device can act on a 2D DNA lattice. Yan also designed a sequence dependent robot. However, this device needs a special platform and needs some extra strands as fuel. In addition to [32], we can find different kinds of robots which are using DNA strands as a fuel. Seeman [63] used DNA strands as a fuel for a walker on a DNA-sequence. This walker used a predefined path. However, Seeman's robot could not walk non-autonomously. This robot can be controlled precisely. Seeman's robot has two feet and can walk forward and backward. This robot does not change the feet order during the walking but there are some other walkers which can change their foot positions during the walking. In both designs, the addition of DNA strands as a fuel helps each foot to attach to the platform, then adding another DNA strand disconnects the other foot and so on. Changing the order of feet can reduce the number of auxiliary DNA strands. To determine the exact position of the walker, usually additional material has been added to the

feet, for example [63] added Psoralen to the end of feet and [64] used fluorescence materials. These additional materials make it easier to monitor to process.

In addition to the walker robots, there are several different non-autonomous devices which can switch between their states. Just like the walkers, these devices need some control mechanisms for each step, and they will produce some waste products. DNA devices can have different number of states. Open/closed states are the most common states in the DNA based devices [65] [77]. Moreover, some of these nanodevices have more than two states. For example, [65] introduced a DNA device which can be in the closed, relaxed or stretched configurations. Just like the DNA walkers, the operation of these devices can be monitored using additional materials such as fluorescence materials.

To compare different non-autonomous nanorobots, we should consider different parameters: number of different states, type of their fuel, number of auxiliary strands as fuel DNA, their dependency to the environment, the amount of energy that they use for the actions, etc. Nonautonomous DNA-based nanorobotics always need control and monitoring by a human or devices. The design of autonomous DNA devices, which can act without assistance of lab procedures, can solve this problem. Although there are various kinds of implementation of autonomous DNA-based devices, those devices usually have a number of limitations. In some cases, they destroy the input after their action or they block the input and the input cannot be used anymore. Moreover, prior implementation of autonomous walkers could walk randomly in forward and backward directions[60]. Recent developed walkers can move in a one-directional path. It is noticeable that one-directional movement means they need fuel, and walking is not reversible anymore. Using DNA hybridization energy as a fuel is one of the basic ideas in this area. Turberfield et al. [37] suggested that hybridization can be used to build an autonomous DNA walker. To reach to this goal, Yin et al [76] experimentally constructed an autonomous walker for the first time. Their walker used alternating actions of restriction enzymes and ligase. Nowadays, there are several implementations of nanodevices which act autonomously. but without any fuel robots have to perform some random actions, like bidirectional walking.

The construction of autonomous DNA-based nanorobots is not restricted to walkers. Mao [68] implemented a nanodevice which continuously changed its state between open and closed.

Moreover, he constructed an autonomous walker which used a different kind of fuel. Their device used a DNAzyme. Using substrate molecules, this DNAzyme constantly extracts chemical energy. Their DNA walker used the energy as the fuel of the motion. Their device also used strand displacement to put its feet on the backbone.

2.4 2D Grammars

In Chapter 3, we propose *self-assembly (SA) hypergraph automata* as a general model for patterned self-assembly and investigate its connections to other models for two-dimensional information and computation, such as 2D (picture) languages and Wang Tile Systems. Although the connection between a specially designed tile self-assembly systems and 2D languages was studied before [26], our proposed automata is the first that studies the connection between the abstract tile assembly model and 2D languages. Here a brief introduction to 2D languages is given, and the relation between 2D languages and self-assembly system are studied in more details in Chapter 3.

Theory of 2D languages is a generalization of the theory of formal languages in one dimension. The first definition of an automata which works on 2D spaces was introduced in 1967 [12]. In 1997, Giammarresi and Resivo [33] defined the family of recognizable picture languages (REC) as a generalization of regular 1D languages.

To understand the theory of picture languages, first of all, a formal definition for a picture is required. Let Σ be a finite alphabet. A picture p over Σ , is a two-dimensional array of letters from Σ . Each of these letters is called a *pixel*. The size of the picture p , denoted by $|p|$, is the pair $(|p|_{row}, |p|_{col})$, where $|p|_{row}$ is the number of rows and $|p|_{col}$ is the number of columns of the picture. Indices grow from top to bottom and from left to right. The set of all pictures over Σ is $\Sigma^{+,+}$. $\Sigma^{*,*}$ is $\Sigma^{+,+} \cup \{\lambda\}$, where λ denotes the empty picture. A 2D language over Σ is a subset of $\Sigma^{*,*}$. Sometimes, the letter $\#$ is used to border the picture. \hat{p} denotes the bordered version of

picture p . For $p \in \Sigma^{k,h}$, \hat{p} is:

$$\hat{p} = \begin{array}{ccccc} \# & \# & \cdots & \# & \# \\ \# & p_{(1,1)} & \cdots & p_{(1,h)} & \# \\ \# & \vdots & \ddots & \vdots & \# \\ \# & p_{(k,1)} & \cdots & p_{(k,h)} & \# \\ \# & \# & \cdots & \# & \# \end{array}$$

Partially bordered pictures [13] have been used in some cases. Since partially bordered pictures are not very common, their description is beyond the scope of this report. Giammarresi and Restivo introduced the family of recognizable picture languages (REC)[33]. In this section, first, the definition of REC using Wang tiles is recalled, and then, an equivalent definition of REC based on the local rules is given [31].

Definition ([25]) A labeled Wang tile, shortly LWT, is a 5-tuple (c_1, c_2, c_3, c_4, a) where for all i , $1 \leq i \leq 4$, c_i belongs to a finite set C of *colours* and a belongs to a finite set Σ of labels.

A Wang tiling system (WTS) is a triple $\langle C, \Sigma, T \rangle$, where $T \subseteq C^4 \times \Sigma$ is a finite set of LWTs. Intuitively, a picture p of size (m, n) over alphabet T is a tiling over T , if for any two adjacent LWTs, the colours of the adjacent borders are the same. For a WTS \mathfrak{T} , $L(\mathfrak{T})$ denotes the language that \mathfrak{T} describes. In other words, $L(\mathfrak{T})$ denotes the set of all pictures that have a valid tiling using T . A language L generated by a WTS is called Wang recognizable.

Exactly the same family of picture languages is introduced by another method using local rules and a projection. In the field of 2D languages, these local rules are called *tiles*. Here the definition of a *tile* is not equal to the definition of a Wang tile.

Definition ([20]) A *tile* is a square picture of size $(2, 2)$. A language $L \subseteq \Sigma^{*,*}$ is local if there exists a finite set Θ of tiles over the alphabet $\Sigma \cup \{\#\}$ such that $L = \{p \in \Sigma^{*,*} \mid [\hat{p}] \subseteq \Theta\}$, where, $[\hat{p}]$ denotes the set of all tiles/subwords of picture \hat{p} of size $(2, 2)$. We will refer to such a language as $LOC(\Theta)$.

Definition ([33]) A tiling system (TS) is the 4-tuple $T = (\Sigma, \Lambda, \Theta, \pi)$, where: Σ and Λ are two finite alphabets, $\pi : \Lambda \rightarrow \Sigma$ is a mapping, Θ is a finite set of 2×2 tiles over the alphabet $\Lambda \cup \{\#\}$.

The language $L(T)$ is the language defined by the tiling system T , if L equals to the language $LOC(\Theta)$ after applying the mapping π on all the pixel of the pictures in $LOC(\Theta)$.

The languages over finite alphabets defined by tiling systems constitute the family REC or TS-recognizable languages on Σ . As previously mentioned, both definitions based on Wang tiles and tiling systems define the same class of languages [25] which is called class of recognizable picture languages. Note that a tiling system without a projection can only recognize a local language. The class of local languages is strictly included in the class of recognizable picture languages. The results in Chapter 3 of this thesis only refer to REC picture languages where the labeled/coloured Wang tiles are used, therefore the same result does not hold for local languages.

All definitions of REC picture languages use non-deterministic behaviours. In [6] definitions of determinism and unambiguity have been introduced.

Definition ([20]) A tiling system $(\Sigma, \Lambda, \Theta, \pi)$ is top-left to bottom-right deterministic (tl2br-deterministic), if for any $\gamma_1, \gamma_2, \gamma_3 \in \Lambda \cup \{\#\}$ and $\sigma \in \Sigma$ there exists at most one tile $t \in \Theta$ with

$$t = \begin{array}{cc} \gamma_1 & \gamma_2 \\ \gamma_3 & \gamma_4 \end{array}, \text{ and } \pi(\gamma_4) = \sigma.$$

A similar definition can be used for unambiguity. The classes of deterministic REC languages and unambiguous REC languages are strictly included in the class of REC. These definition implicitly restrict the order that pixels of a picture are read. For example, if the a top-left to bottom-right deterministic system is used, the pixel at position (x, y) has to be read after the pixel in the position (x', y') , assuming that $x > x'$ and $y > y'$. In these kind of tiling system the definitions of determinism and unambiguity are dependent on the reading path [6]. In other words, for different reading strategies, different classes of languages will be defined [47]. A tiling system that is top-left to bottom-right can only scan the given picture on a path that goes from left to right and from top to bottom, and it can never scan the picture in other directions. Tiling systems have become an accepted model for 2D languages. However, if we want to define an automaton based on them an explicit definition of scanning path is required. A definition for a tiling automaton has been introduced in [7]. This automaton uses a scanning strategy. For example, scanning strategy in tiling automata can depend on the size of the picture. For any scanning strategy, there exists a corresponding different class in Chomsky's hierarchy. Tiling

automata are not the first automata for picture languages. Before tiling automata, 4FA had been introduced [12].

Definition ([33]) A 4FA is a 7-tuple $A = (\Sigma, Q, \{t, b, l, r\}, q_0, q_a, q_r, \delta)$, where Σ is the input alphabet, Q is the set of states, q_0, q_a, q_r are three distinguished states, called initial, accepting and rejecting states, $\delta : (Q/q_a, q_r) \times \Sigma \rightarrow 2^{(Q \times \{t, b, l, r\})}$ is the transition function. t, b, l , and r represent the directions.

A 4FA automaton has a head. If the automaton is in the state $q \in Q/q_a, q_r$ and the head is above a pixel $\sigma \in \Sigma$, based on the transition function δ , automaton will change the state and move to one of the neighbour pixels. The 4FA can pass each of the pixels of a given picture more than once. Moreover, 4FA can leave some of the pixels unread. Since the definition of 4FA is not similar to the self-assembly process, it cannot be an interesting tool for biological computations. In contrast, tiling automata work on picture languages using only local rules, and they pass every pixel exactly one time. A tiling automaton starts with a picture and in each step tries to replace one of the letters with one of the states.

The tl2br tiling automata (defined below) assume that tiling operation will start from top-left and will continue to bottom-right. A tiling automaton needs an explicit definition of a path to find the next position of its head. The path in a tl2br tiling automaton starts from top-left, goes through each row from left to right and then scan the next row starting from the left column. An informal formal definition of tiling automata is as follows. A tiling automaton [7]) of type tl2br is consists of a tiling system, a tl2br-directed scanning strategy, a two dimensional data structure to keep some temporary states, and a transition function. A tl2br tiling automaton starts from the top left pixel of the given picture, and in each transition reads the state of the corresponding pixels of the top, left, and top-left pixels of the current head position. Based on these states, tiling automaton changes the state corresponding the pixel under the head. Afterwards, the heads moves to the next pixel following the scanning strategy. Here the scanning strategy must be defined independently from the content of the picture.

Although tiling automata are more similar to the self-assembly process, dependence of the scanning strategy on the size of the picture is a restriction that is hard to simulate in self-

assembly systems. Recently, Lonati and Pradella [48] introduced a new automaton based on Wang tiles. A Wang automaton has an explicit definition of a strategy path. However, the strategy path in Wang automata has to be *polite*. *Polite* strategies do not have any information about the size of the picture; they are only sensitive to the borders. A Wang automaton starts from a picture, then in each step colours around one of the letters will be defined, and the coloured area will be extended. Using Wang tiles and also polite scanning strategies made Wang automata an interesting device for self-assembly modelling. However, all existing tiling systems that were discussed in the chapter including Wang automata still depend on the definition of scanning strategy [14, 48, 33]. Moreover, polite scanning strategies are less powerful than regular scanning strategies. Lonati and Pradella [49] compared possible scanning strategies and their powers, and they proved that Wang automata cannot simulate all the scanning strategies [14].

Bibliography

- [1] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024, 1994.
- [2] L. Adleman, Q. Cheng, A. Goel, M.-D. Huang, D. Kempe, P. M. de Espanés, and P. W. K. Rothmund. Combinatorial optimization problems in self-assembly. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 23–32, 2002.
- [3] L. M. Adleman. Toward a mathematical theory of self-assembly. Department of Computer Science, University of Southern California, 1999.
- [4] L. M. Adleman, Q. Cheng, A. Goel, and M. A. Huang. Running time and program size for self-assembled squares. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing*, pages 740–748, 2001.
- [5] G. Aggarwal, M. H. Goldwasser, M.-Y. Kao, and R. T. Schweller. Complexities for

- generalized models of self-assembly. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '04, pages 880–889, 2004.
- [6] M. Anselmo, D. Giammarresi, and M. Madonia. From determinism to non-determinism in recognizable two-dimensional languages. In *Developments in Language Theory, 11th International Conference, DLT*, pages 36–47. 2007.
- [7] M. Anselmo, D. Giammarresi, and M. Madonia. Tiling automaton: A computational model for recognizable two-dimensional languages. In *Implementation and Application of Automata, 12th International Conference, CIAA*, pages 290–302. 2007.
- [8] R. D. Barish, P. W. K. Rothmund, and E. Winfree. Two computational primitives for algorithmic self-assembly: copying and counting. *Nano Letters*, 5(12):2586–2592, 2005.
- [9] R. D. Barish, R. Schulman, P. Rothmund, and E. Winfree. An information-bearing seed for nucleating algorithmic self-assembly. *Proceedings of the National Academy of Sciences of the United States of America*, 106:6054–6059, 2009.
- [10] R. Berger. *Undecidability of the Domino Problem*. Memoirs of the American Mathematical Society. 1966.
- [11] G. H. Bernstein. Quantum-dot cellular automata: computing by field polarization. In *Proceedings of the 40th Design Automation Conference, DAC 2003*, pages 268–273.
- [12] M. Blum and C. Hewitt. Automata on a 2-dimensional tape. In *8th Annual Symposium on Switching and Automata Theory*, pages 155–160, 1967.
- [13] P. Bonizzoni, C. Ferretti, A. R. Sagaya Mary, and G. Mauri. Picture languages generated by assembling tiles. In *Language and Automata Theory and Applications, Third International Conference, LATA*, pages 224–235, 2009.
- [14] R. Brijder and H. J. Hoogeboom. Perfectly quilted rectangular snake tilings. *Theor. Comput. Sci.*, 410(16):1486–1494, 2009.
- [15] Y. Brun. Efficient 3-sat algorithms in the tile assembly model. *Natural Computing*, 11(2):209–229, 2012.

- [16] N. Bryans, E. Chiniforooshan, D. Doty, L. Kari, and S. Seki. The power of nondeterminism in self-assembly. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 590–602, 2011.
- [17] H. Bui, C. Onodera, C. Kidwell, Y. Tan, E. Graugnard, W. Kuang, J. Lee, W. B. Knowlton, B. Yurke, and W. L. Hughes. Programmable periodicity of quantum dot arrays with DNA origami nanotubes. In *Nano Letters*, volume 10, pages 3367–3372, 2010.
- [18] H.-L. Chen and A. Goel. Error free self-assembly using error prone tiles. In *DNA Computing*, volume 3384 of *Lecture Notes in Computer Science*, pages 62–75. 2005.
- [19] H.-L. Chen, A. Goel, and C. Luhrs. Dimension augmentation and combinatorial criteria for efficient error-resistant DNA self-assembly. In *Proceedings of the nineteenth annual ACM-SIAM symposium on discrete algorithms, SODA '08*, pages 409–418, 2008.
- [20] A. Cherubini and M. Pradella. Picture languages: From Wang tiles to 2D grammars. In *Algebraic Informatics*, volume 5725 of *Lecture Notes in Computer Science*, pages 13–46. 2009.
- [21] B. S. Chlebus. Domino-tiling games. *J. Comput. Syst. Sci.*, 32(3):374–392, 1986.
- [22] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. *ACM Trans. Graph.*, 22(3):287–294, 2003.
- [23] M. Cook, P. W. K. Rothmund, and E. Winfree. Self-assembled circuit patterns. In *DNA Computing, 9th International Workshop on DNA Based Computers, DNA9*, pages 91–107, 2003.
- [24] K. Culik II. An aperiodic set of 13 Wang tiles. *Discrete Mathematics*, 160:245–251, 11 1996.
- [25] L. De Prophetis and S. Varricchio. Recognizability of rectangular pictures by Wang systems. *J. Autom. Lang. Comb.*, 2:269–288, July 1998.
- [26] E. Dolzhenko, N. Jonoska, and N. C. Seeman. Transducer generated arrays of robotic nano-arms. *Natural Computing*, 9(2):437–455, 2010.

- [27] D. Doty. Randomized self-assembly for exact shapes. *SIAM J. Comput.*, 39(8):3521–3552, 2010.
- [28] D. Doty, L. Kari, and B. Masson. Negative interactions in irreversible self-assembly. In *Algorithmica*, volume 66, pages 153–172, 2013.
- [29] D. Doty, M. J. Patitz, D. Reishus, R. T. Schweller, and S. M. Summers. Strong fault-tolerance for self-assembly with fuzzy temperature. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 417–426, 2010.
- [30] D. Doty, M. J. Patitz, and S. M. Summers. Limitations of self-assembly at temperature 1. *Theor. Comput. Sci.*, 412(1-2):145–158, 2011.
- [31] B. Durand, L. Levin, and A. Shen. Local rules and global order, or aperiodic tilings. *The Mathematical Intelligencer*, 27:64–68, 2005.
- [32] L. Feng, S. H. Park, J. H. Reif, and H. Yan. A two-state DNA lattice switched by DNA nanoactuator. *Angewandte Chemie*, 2005.
- [33] D. Giammarresi and A. Restivo. Two-dimensional languages. *Handbook of formal languages*, vol. 3, pages 215–267, 1997.
- [34] K. Gilpin and D. Rus. A distributed algorithm for 2d shape duplication with smart pebble robots. In *Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2012*, pages 3285–3292, 2012.
- [35] E. Goles and I. Rapaport. Complexity of tile rotation problems. *Theor. Comput. Sci.*, 188(1-2):129–159, 1997.
- [36] E. Goles and I. Rapaport. Tiling allowing rotations only. *Theor. Comput. Sci.*, 218(2):285–295, 1999.
- [37] S. J. Green, J. Bath, and A. J. Turberfield. Coordinated chemomechanical cycles: A mechanism for autonomous molecular motion. *Phys. Rev. Lett.*, 101(23):238101, 2008.
- [38] Y. S. Gurevich and I. O. Koryakov. Remarks on Berger’s paper on the domino problem. *Siberian Mathematical Journal*, 13:319–321, 1972.

- [39] S. Jaswinder, C. Rahul, L. Yan, K. Yonggang, and Y. Hao. DNA-templated self-assembly of two-dimensional and periodical gold nanoparticle arrays. *Angewandte Chemie International Edition*, 45(5):730–735, 2006.
- [40] N. Jonoska and G. L. McColm. Flexible versus rigid tile assembly. In *Unconventional Computation, 5th International Conference*, pages 139–151, 2006.
- [41] M. Kao and R. T. Schweller. Reducing tile complexity for self-assembly through temperature programming. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 571–580, 2006.
- [42] M.-Y. Kao and R. Schweller. Randomized self-assembly for approximate shapes. In *Automata, Languages and Programming*, volume 5125 of *Lecture Notes in Computer Science*, pages 370–384. 2008.
- [43] J. Kari. A small aperiodic set of Wang tiles. *Discrete Mathematics*, 160:259–264, 1996.
- [44] L. Kari, S. Seki, and Z. Xu. Triangular and hexagonal tile self-assembly systems. In *Computation, Physics and Beyond - International Workshop on Theoretical Computer Science, WTCS*, pages 357–375, 2012.
- [45] T. H. LaBean, E. Winfree, and J. H. Reif. Experimental progress in computation by self-assembly of DNA tilings. In *DNA Based Computers, Proceedings of a DIMACS Workshop*, pages 123–140, 1999.
- [46] C. Lin, Y. Liu, S. Rinker, and H. Yan. DNA tile based self-assembly: Building complex nanoarchitectures. *ChemPhysChem*, 7(8):1641–1647, 2006.
- [47] V. Lonati and M. Pradella. Snake-deterministic tiling systems. In *Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science*, pages 549–560, 2009.
- [48] V. Lonati and M. Pradella. Picture recognizability with automata based on Wang tiles. In *SOFSEM 2010: 36th Conference on Current Trends in Theory and Practice of Computer Science*, pages 576–587. 2010.

- [49] V. Lonati and M. Pradella. Strategies to scan pictures with automata based on Wang tiles. *RAIRO - Theor. Inf. and Applic.*, 45(1):163–180, 2011.
- [50] X. Ma and F. Lombardi. Synthesis of tile sets for DNA self-assembly. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(5):963–967, 2008.
- [51] C. Moore, I. Rapaport, and E. Rémila. Tiling groups for Wang tiles. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 402–411, 2002.
- [52] C. T. O’Mahony, R. A. Farrell, T. Goshal, J. D. Holmes, and M. A. Morris. The thermodynamics of defect formation in self-assembled systems. In *Thermodynamics - Systems in Equilibrium and Non-Equilibrium*, pages 279–307. 2011.
- [53] J. Padilla, R. Sha, M. Kristiansen, J. Chen, N. Jonoska, and N. Seeman. A signal-passing DNA-strand-exchange mechanism for active self-assembly of DNA nanostructures. *Angewandte Chemie International Edition*, 54:5939–5942, 2015.
- [54] M. J. Patitz, R. T. Schweller, and S. M. Summers. Exact shapes and Turing universality at temperature 1 with a single negative glue. In *DNA Computing and Molecular Programming - 17th International Conference, DNA 17*, pages 175–189, 2011.
- [55] J. Reif, S. Sahu, and P. Yin. Compact error-resilient computational DNA tilings. In *Nanotechnology: Science and Computation*, Natural Computing Series, pages 79–103. 2006.
- [56] J. H. Reif, T. H. LaBean, S. Sahu, H. Yan, and P. Yin. Design, simulation, and experimental demonstration of self-assembled DNA nanostructures and motors. In *Unconventional Programming Paradigms, International Workshop UPP*, pages 173–187, 2004.
- [57] P. W. K. Rothmund, A. Ekani-Nkodo, N. Papadakis, A. Kumar, D. K. Fygenson, and E. Winfree. Design and characterization of programmable DNA nanotubes. *Journal of the American Chemical Society*, 126(50):16344–16352, 2004.

- [58] P. W. K. Rothemund, N. Papadakis, and E. Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biol*, 2(12):e424, 12 2004.
- [59] P. W. K. Rothemund and E. Winfree. The program-size complexity of self-assembled squares (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 459–468, 2000.
- [60] S. Sahu. *DNA-Based Self-Assembly and Nanorobotics: Theory and Experiments*. PhD thesis, Department of Computer Science, Duke University, 2007.
- [61] R. Schulman and E. Winfree. Programmable control of nucleation for algorithmic self-assembly. In *DNA Computing*, volume 3384 of *Lecture Notes in Computer Science*, pages 319–328. 2005.
- [62] N. C. Seeman. Nucleic acid junctions and lattices. *Journal of Theoretical Biology*, 99(2):237 – 247, 1982.
- [63] W. B. Sherman and N. C. Seeman. A precisely controlled DNA biped walking device. *Nano Letters*, 4(7):1203–1207, 2004.
- [64] J.-S. Shin and N. A. Pierce. A synthetic DNA walker for molecular transport. *Journal of the American Chemical Society*, 126(35):10834–10835, 2004.
- [65] F. C. Simmel and B. Yurke. A DNA-based molecular device switchable between three distinct mechanical states. *Applied Physics Letters*, 80(5):883–885, 2002.
- [66] D. Soloveichik and E. Winfree. Complexity of self-assembled shapes. In *DNA Computing, 10th International Workshop on DNA Computing, DNA 10*, pages 344–354, 2004.
- [67] G. Terrazas, M. Gheorghe, G. Kendall, and N. Krasnogor. Evolving tiles for automated self-assembly design. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC*, pages 2001–2008, 2007.
- [68] Y. Tian, Y. He, Y. Chen, P. Yin, and C. Mao. A DNzyme that walks processively and autonomously along a one-dimensional track. *Angewandte Chemie*, 117(28):4429–4432, 2005.

- [69] H. Wang. Proving theorems by pattern recognition I. *Commun. ACM*, 3(4):220–234, 1960.
- [70] E. Winfree. On the computational power of DNA annealing and ligation. In *DNA Based Computers, Proceedings of a DIMACS series in Discrete Mathematics and Theoretical Computer Science. No.27. American Mathematical Society*, pages 199–221, 1995.
- [71] E. Winfree. *Algorithmic Self-assembly of DNA*. PhD thesis, California Institute of Technology, 1998.
- [72] E. Winfree. Simulations of computing by self-assembly. Technical report, California Institute of Technology, 1998.
- [73] E. Winfree and R. Bekbolatov. Proofreading tile sets: Error correction for algorithmic self-assembly. In *DNA Computing*, volume 2943 of *Lecture Notes in Computer Science*. 2004.
- [74] E. Winfree, F. Liu, L. A. Wenzler, and N. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394:539–44, 1998.
- [75] C. Worman and M. D. Watson. Tiling layouts with dominoes. In *Proceedings of the 16th Canadian Conference on Computational Geometry, CCCG'04*, pages 86–90, 2004.
- [76] P. Yin, H. Yan, X. G. Daniell, A. J. Turberfield, and J. H. Reif. A unidirectional DNA walker that moves autonomously along a track. *Angewandte Chemie International Edition*, 43(37):4906–4911, 2004.
- [77] B. Yurke, A. J. Turberfield, A. P. Mills, F. C. S. Jr, and J. L. Neumann. A DNA-fuelled molecular machine made of DNA. *Nature*, 406:605–608, 2000.

Chapter 3

Hypergraph Automata: A Theoretical Model for Patterned Self-assembly ¹

3.1 Introduction

DNA-based self-assembly is an autonomous process whereby a disordered system of DNA sequences forms an organized structure or pattern as a consequence of Watson-Crick complementarity of DNA sequences, without external direction. A DNA-tile-based self-assembly system starts from DNA “tiles”, each of which is formed beforehand from carefully designed single-stranded DNA sequences which bind via Watson-Crick complementarity and ensure the tiles’ shape (square) and structure. In particular, the sides and interior of the square are double-stranded DNA sequence, while the corners have protruding DNA single strands that act as “sticky ends”. Subsequently, the individual tiles are mixed together and interact locally via their sticky-ends to form DNA-based *supertiles* whose structure is dictated by the base-composition of the individual tiles’ sticky ends. Winfree [20] introduced the abstract Tile Assembly Model (aTAM) as a mathematical model for tile-based self-assembly systems. Ma and Lombardi [17] introduced the patterned self-assembly of single patterns, whereby coloured tiles self-assemble to build a particular rectangular *coloured pattern*. Patterned self-assembly models a particular type of application in which tiles may differ from each other by some dis-

¹This chapter is based on the paper Lila Kari, Steffen Kopecki, Amirhossein Simjour: Hypergraph Automata: a Theoretical Model for Patterned Self-assembly. Int. J. Found. Comput. Sci. 25(4): 419-440 (2014)

tinguishable properties, modelled as colours [19, 2]. Orponen et al. [9, 14] designed several algorithms to find the minimum tile set required to construct one given coloured pattern. Czeizler and Popa [6] proved that this minimization problem is NP-hard. The problem remains NP-hard for patterns with a constant number of 29 colours [12] and for three-coloured patterns when the tile numbers for two of the three colours are fixed [13].

In this paper, we propose *self-assembly (SA) hypergraph automata* as a general model for patterned self-assembly and investigate its connections to other models for two-dimensional information and computation, such as 2D (picture) languages and Wang Tile Systems. A 2D (picture) language consists of 2D words (pictures), defined as mappings $p : [m] \times [n] \rightarrow [k]$ from the points in the two-dimensional space to a finite alphabet of cardinality k . Here, $[k]$ denotes the set $[k] = \{1, 2, \dots, k\}$. Note that, if we take the alphabet $[k]$ to be a set of colours, the definition of a picture is analogous to that of a coloured pattern [17].

Early generating/accepting systems for 2D languages comprise 2×2 tiles [8], 2D automata [3], two-dimensional on-line tessellation acceptors [10], and 2D grammars [5]. More recently a generating system was introduced by Prophetis and Varricchio [7] that used *Wang tiles*. A *Wang tile system* [7] is a specialized tile-based model that generates the class of *recognizable picture languages*, a subclass of the family of 2D languages. The class of recognizable picture languages is also accepted by *Wang automata*, a model introduced in [15]. Like other automata for 2D languages [1], Wang tile automata use an explicit pre-defined scanning strategy [16] when reading the input picture and the accepted language depends on the scanning strategy that is used. Due to this, Wang automata are a suboptimal model for self-assembly. Indeed, if we consider the final supertile as given, the order in which tiles are read is irrelevant. On the other hand, if we consider the self-assembly process which results in the final supertile, an “order of assembly” cannot be pre-imposed. In contrast to Wang automata, SA-hypergraph automata are scanning-strategy-independent. In fact, Wang tile systems only can be used for self-assembly systems in temperature 1, but SA-hypergraph automata does not have this restriction. Moreover, SA-hypergraph automata can be applied to non-rectangular shapes, which are out of the scopes of the Wang tile systems and Wang tile automata.

SA-hypergraph automata are a modification of the hypergraph automata introduced by Janssens and Rozenberg [11] in 1982. An SA-hypergraph automaton (Section 3.3) accepts

a language of labelled “rectangular grid graphs”, wherein the labels are meant to capture the notion of colours used in patterned self-assembly. An SA-hypergraph automaton consists of an underlying labelled graph (labelled nodes and edges) and a set of *hyperedges*, each of which is a subset of the set of nodes of the underlying graph. Intuitively, the hyperedges are meant to model tiles or supertiles while the underlying graph describes how these can attach to each other, similar to a self-assembly process.

We investigate the computational power of SA-hypergraph automata and prove that for every recognizable picture language L there is an SA-hypergraph automaton that accepts L (Thm. 3.4.1). Moreover, we prove that for any restricted SA-hypergraph automaton, there exists a Wang tile system that accepts the same language of coloured patterns (Thm. 3.4.2). Here, restricted SA-hypergraph automaton means an SA-hypergraph automaton in which certain situations that cannot occur during self-assembly are explicitly excluded.

3.2 Preliminaries

A picture (two-dimensional word) p over the alphabet Σ is a two dimensional matrix of letters from Σ . Each element of this matrix is called a pixel. $p_{(i,j)}$ denotes the pixel in the i th row and j th column of this matrix. Two pixels $p_{(i,j)}$ and $p_{(i',j')}$ are adjacent if $|i - i'| + |j - j'| = 1$. The function $w(p)$ denotes the width and $h(p)$ denotes the height of the picture p . Σ^{**} is the set of all pictures over the alphabet Σ . Let $\#$ be a letter which does not belong to the alphabet Σ . The *framed picture* \hat{p} of $p \in \Sigma^{**}$ is defined as:

$$\hat{p} = \begin{array}{ccccc} \# & \# & \cdots & \# & \# \\ \# & p_{(1,1)} & \cdots & p_{(1,w(p))} & \# \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \# & p_{(h(p),1)} & \cdots & p_{(h(p),w(p))} & \# \\ \# & \# & \cdots & \# & \# \end{array}$$

A *picture language* (2D language) is a set of pictures over an alphabet Σ . For example, $L = \{p \in \Sigma^{**} \mid \text{for all } 1 \leq i \leq h(p), p_{(i,1)} = p_{(i,w(p))}\}$ is the language of all rectangles that have the same first and last column.

A function $\delta: \mathbb{N}^2 \rightarrow \mathbb{N}^2$ is a *translation function* if there exists $i', j' \in \mathbb{Z}$ such that $\delta(i, j) = (i + i', j + j')$ for all $i, j \in \mathbb{N}$. A *subpicture* over Σ is a two-dimensional matrix of letters from $\Sigma \cup \{\text{empty}\}$. A subpicture q is *connected* if for every pair of pixels $q_{(i',j')}, q_{(i,j)} \in \Sigma$ there exists a sequence of pixels $s = \langle s_0, s_1, \dots, s_n \rangle$ from q such that $s_0 = q_{(i,j)}$ and $s_n = q_{(i',j')}$, for all $0 \leq k < n$, we have $s_k \in \Sigma$. Moreover, s_k and s_{k+1} must be adjacent. If p is a picture, then q is a subpicture of p if there exists a translation function δ such that for all $(i, j) \in [h(q)] \times [w(q)]$ we have either $q_{(i,j)} = \text{empty}$ or $q_{(i,j)} = p_{\delta(i,j)}$.

A *picture tile* is a 2×2 picture (for example

a	b
c	d

). The language defined by a set of picture tiles Δ over the alphabet $\Sigma \cup \{\#\}$ is denoted by $\mathcal{L}(\Delta)$ and is defined as the set of all pictures $p \in \Sigma^{**}$ such that any 2×2 subpicture of \hat{p} is in Δ . Giammarresi and Restivo [8] defined a *Picture Tiling System* (PTS) as a 4-tuple $T = (\Sigma, \Gamma, \Delta, \pi)$, where Σ and Γ are two finite alphabets, Δ is a finite set of picture tiles over $\Gamma \cup \{\#\}$ and $\pi: \Gamma \rightarrow \Sigma$ is a projection. The PTS T recognizes the language $\mathcal{L}(T) = \pi(\mathcal{L}(\Delta))$. A picture language L is called *PTS-recognizable* if there exists a picture tiling system T such that $L = \mathcal{L}(T)$. Tiling systems have also been studied in conjunction with rational graphs, as tools for recognizing one-dimensional context-sensitive languages [4] [18].

An equivalent definition of recognizability was proposed using labelled Wang tiles [16]. A labelled Wang tile, shortly LWT, is a labelled unit square whose edges may be coloured. Formally, a LWT is a 5-tuple (c_N, c_E, c_S, c_W, l) , where l belongs to a finite set of labels Σ and c_N, c_E, c_S , and c_W belong to $C \cup \{\#\}$ where C is a finite set of colours and $\#$ represents an uncoloured edge. Intuitively, c_N, c_E, c_S , and c_W represent the colour of the north, east, south, and west edge of the tile, respectively. Labelled Wang tiles cannot rotate. The colours on the north, south, east, and west edges of an LWT t are denoted by $\sigma_N(t)$, $\sigma_S(t)$, $\sigma_E(t)$, and $\sigma_W(t)$, respectively; moreover, $\lambda(t)$ denotes the label of t .

A Wang Tile System (WTS)[7] is a triple $W = (\Sigma, C, \Theta)$ where Σ and C are two finite alphabets (the alphabet of tile labels and the alphabet of colours, respectively) with $\# \notin C$, and Θ is a finite set of labelled Wang tiles with labels from Σ and colours from C . The WTS W recognizes the picture language $\mathcal{L}(W)$ where the picture $p \in \Sigma^{**}$ belongs to $\mathcal{L}(W)$ if and only if there exists a mapping $m: [h(p)] \times [w(p)]$ from the pixels of p to tiles from Θ such

that the label of the tile $m(p_{(i,j)})$ is equal to $p_{(i,j)}$; moreover, this mapping must be *mismatch free*. The mapping m is mismatch free if for two adjacent pixels $p_{(i,j)}$ and $p_{(i+1,j)}$ in p the south edge of $m(p_{(i,j)})$ and the north edge of $m(p_{(i+1,j)})$ are coloured by the same colour from C ; for two adjacent pixels $p_{(i,j)}$ and $p_{(i,j+1)}$ in p the east edge of $m(p_{(i,j)})$ and the west edge of $m(p_{(i,j+1)})$ are coloured by the same colour from C ; and for every border pixel $p_{(i,j)}$ with $i = 1$, $j = 1$, $i = h(p)$, or $j = w(p)$ we require that the north, west, south, or east edge, respectively, of $m(p_{(i,j)})$ is uncoloured. For a pixel in a corner, e. g. $p_{(1,1)}$, this implies that two edges are uncoloured. Let \bar{p} be a two dimensional array of labelled Wang tiles from Θ . We call \bar{p} a Wang tiled version of the picture p if the width and the height of p and \bar{p} are equal, and there exists a mismatch free mapping m such that for any i and j we have $\bar{p}_{(i,j)} = m(p_{(i,j)})$. Two tiles $\bar{p}_{(i,j)}$ and $\bar{p}_{(i',j')}$ are adjacent if the pixels $p_{(i,j)}$ and $p_{(i',j')}$ are adjacent. A language L is *WTS-recognizable* if there exists a Wang tile system W such that W recognizes L . Figure 3.1 shows an example.

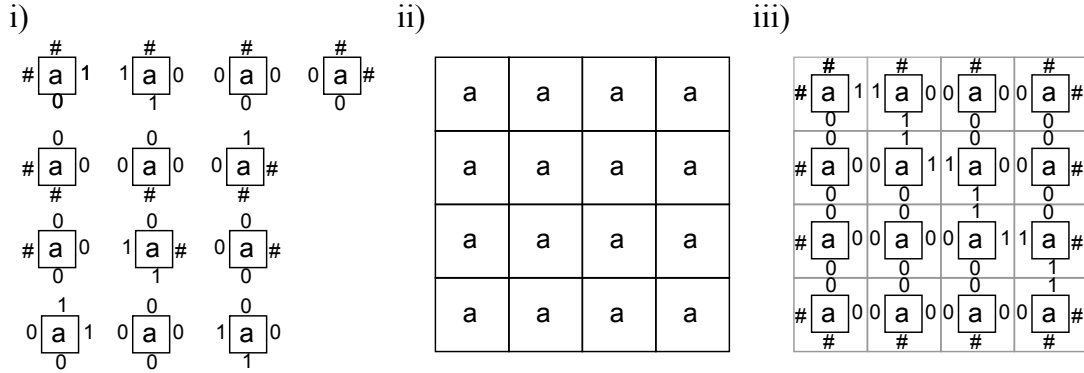


Figure 3.1: Let $W = (\Sigma, C, \Theta)$ be the Wang Tile System where $\Sigma = \{a\}$, $C = \{0, 1\}$ and Θ consists of the 13 LWTs shown in i). This Wang tile system recognizes the picture language containing all square pictures p with $h(p) = w(p) \geq 3$ and where every pixel is labelled by a . Part ii) is an example picture and iii) shows the Wang tiled version of the picture in part ii).

Proposition 3.2.1 ([8]) *A picture language L is PTS-recognizable if and only if it is WTS-recognizable.*

A coloured pattern, as defined in [17] is the end result of a self-assembly process that starts with a fixed-size L -shaped seed supertile and proceeds as in Figure 3.2, (i), until one coloured rectangle is formed. Note that Wang Tile Systems can be seen as generating (potentially infinite) languages of such coloured patterns where the L -shaped seed is of an arbitrary size and

is generated starting from a single-tiled seed with uncoloured North and West edges, and is extended by tiles with uncoloured North or West edges, as shown in Figure 3.2, (ii).

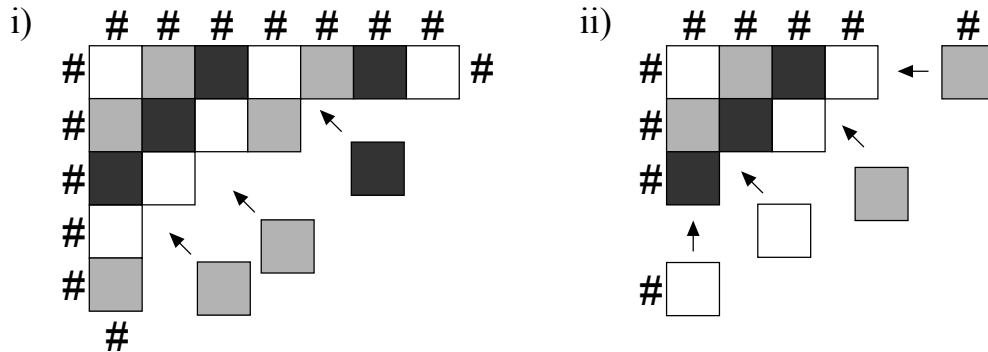


Figure 3.2: (i) The self-assembly of a single *coloured pattern*, starting with a fixed-size *L-shaped seed*. (ii) The process of generating a *picture* in the language of a *Wang Tile System*.

3.3 Hypergraph Automata

Let $f: A \rightarrow B$ be a function and let $A' \subseteq A$. The *restriction* of f to A' is $f|_{A'}: A' \rightarrow B$ such that $f|_{A'}(x) = f(x)$ for all $x \in A'$. For any set A we let $id_A: A \rightarrow A$ denote the *identity*. When the set A is clear from the context, we will omit the subscript and simply write id .

Let Σ be an alphabet. A *pseudo-picture graph* is a directed labelled graph $G = (N, E_v \cup E_h, \pi)$ where N is a finite set of nodes, $E_v, E_h \subseteq N \times N$ are two sets of edges such that $E_v \cap E_h = \emptyset$, and $\pi: N \rightarrow \Sigma$ is the label function. Edges from E_v and E_h will frequently be denoted by \xrightarrow{v} and \xrightarrow{h} , respectively. The *node-induced subgraph* of G by a subset $N' \subseteq N$ is defined as the graph $(N', E'_v \cup E'_h, \pi|_{N'})$ where $E'_v = \{(x, y) \in E_v \mid x, y \in N'\}$ and $E'_h = \{(x, y) \in E_h \mid x, y \in N'\}$. A graph G' is called a *full subgraph* of G if for some $N' \subseteq N$ it is the node-induced subgraph of G by N' .

A pseudo-picture graph $G = (N, E_v \cup E_h, \pi)$ is an $(n \times m)$ -*picture graph* (for $n, m \in \mathbb{N}$) if there is a bijection $f_G: N \rightarrow [n] \times [m]$ such that for $x, y \in N$, we have $(x, y) \in E_v$ if and only if $f_G(x) + (1, 0) = f_G(y)$, and $(x, y) \in E_h$ if and only if $f_G(x) + (0, 1) = f_G(y)$. We want to stress that we do not use Cartesian coordinates; our pictures are defined as matrices, hence, incrementing the first coordinate corresponds to a step downwards, and incrementing

the second coordinate corresponds to a step rightwards. In other words, the nodes of a picture graph G can be embedded in \mathbb{N}^2 such that every edge in E_v has length 1 and points downwards, every edge in E_h has length 1 and points rightwards, and every two nodes with Euclidean distance 1 are connected by an edge. Note that if a pseudo-picture graph is an $n \times m$ -picture graph, it cannot be an $n' \times m'$ -picture graph with $n \neq n'$ or $m \neq m'$, and the function f_G is unique. If G is a picture graph, we call $e \in E_v$ a *vertical edge* and $e \in E_h$ a *horizontal edge*. The set of all picture graphs is denoted by \mathcal{G} . Every $n \times m$ -picture graph $G = (N, E_v \cup E_h, \pi)$ represents a picture $p(G) \in \Sigma^{**}$ with $h(p(G)) = n$ and $w(p(G)) = m$. More precisely, for all $(i, j) \in [n] \times [m]$ we let $p(G)_{(i,j)} = \pi(f_G^{-1}(i, j))$. Hence, $p: \mathcal{G} \rightarrow \Sigma^{**}$ can be seen as a function.

A connected pseudo-picture graph G is called a *subgrid* if it is a full subgraph of a picture graph G' . We also say G is a subgrid of G' .

A *hypergraph* [11] is a triple $H = (N, E, f)$ where N is the finite set of nodes, E is the finite set of *hyperedges*, and $f: E \rightarrow \mathcal{P}(N)$ is a function assigning to each hyperedge a set of nodes; the same set of nodes may be assigned to two distinct hyperedges. For every hyperedge $e \in E$, we let

$$I_H(e) = \{x \in N \mid \exists e' \in E \setminus \{e\}: x \in f(e) \cap f(e')\}$$

be the set of *intersecting nodes* in $f(e)$. Rozenberg and Janssens [11] introduced *hypergraph automata* to describe graph languages. Here, we modified their definition in order to study pseudo-picture graphs. The formal definition is as follows.

Definition A *self-assembly (SA) hypergraph automaton* is a tuple $A = (N, E, f, d, G, E_0)$ where $H = (N, E, f)$ is a hypergraph, called the *underlying hypergraph*, $d: E \rightarrow I_H \times I_H$ is the *transition function* assigning to each hyperedge $e \in E$ a transition $Q_1 \rightarrow Q_2$ with $Q_1, Q_2 \subseteq I_H(e)$, G is a pseudo-picture graph with node set N called the *underlying graph*, and $E_0 \subseteq E$ is the set of *initial hyperedges*.

Every hyperedge $e \in E$ defines a graph G_e which is the subgraph of G induced by $f(e)$. For $d(e) = Q_1 \rightarrow Q_2$ we call Q_1 and Q_2 the *incoming active nodes* and *outgoing active nodes* of G_e , respectively. In order for the hypergraph automaton to be well-defined, we require that G_e is connected and that the subgraph of G_e induced by its incoming active nodes is also connected, for all $e \in E$. If $e \in E_0$, then G_e is also called an *initial graph*.

A *configuration* of the hypergraph automaton A is a triple (M, O, g) where $M = (N_M, E_{M,v} \cup E_{M,h}, \pi_M)$ is a subgrid, $O \subseteq N_M$ is the set of *active nodes*, and $g: N_M \rightarrow N$ is a function such that $\pi_M(x) = \pi(g(x))$ for all $x \in N_M$. The set N_M consists of (possibly multiple) copies of nodes from N and the function g assigns to each node in N_M its original node in N . An edge $(x, y) \in E_{M,h}$ is a copy of the edge $(g(x), g(y)) \in E_h$ and $(x, y) \in E_{M,v}$ is a copy of the edge $(g(x), g(y)) \in E_v$. However, for two nodes x and y in M , if their originals $g(x)$ and $g(y)$ are connected by a horizontal (or vertical) edge, this does not imply that x and y are connected by a horizontal (or vertical) edge.

Let (M_1, O_1, g_1) be a configuration with $M_1 = (N_1, E_{1,v} \cup E_{1,h}, \pi_1)$ and let $e \in E$ be a hyperedge with $d(e) = Q_1 \rightarrow Q_2$. If there exists a non-empty subset $P \subseteq O_1$ such that $g_1|_P$ forms a graph-isomorphism from the subgraph of M_1 induced by P to the subgraph of G_e induced by the incoming active nodes Q_1 , then the hyperedge e defines a *transition* or *derivation step* $(M_1, O_1, g_1) \xrightarrow{A} (M_2, O_2, g_2)$. Informally speaking, the resulting graph M_2 consists of joining together the graphs M_1 and G_e by identifying every node $x \in P$ with the corresponding node $g_1(x) \in Q_1$. The active nodes O_2 in M_2 are the active nodes $O_1 \setminus P$ in M_1 plus the outgoing active nodes Q_2 in G_e , see Figure 3.3. We also say that (M_2, O_2, g_2) is the result of *gluing* the hyperedge e to (M_1, O_1, g_1) . Formally, the configuration (M_2, O_2, g_2) where $M_2 = (N_2, E_{2,v} \cup E_{2,h}, \pi_2)$ is constructed as follows. Let $N' = \{x' \mid x \in f(e) \setminus Q_1\}$ be a set containing a copy of each node from G_e except for the incoming active nodes such that $N' \cap N_1 = \emptyset$, where x' is defined as a copy of node x . Let $N_2 = N_1 \cup N'$ and let $g_2: N_2 \rightarrow N$ be such that $g_2(x) = g_1(x)$ for $x \in N_1$ and $g_2(x') = x$ for $x' \in N'$. An edge (x, y) belongs to $E_{2,v}$ if $(x, y) \in E_{1,v}$ or $x, y \in P \cup N'$ and $(g_2(x), g_2(y)) \in E_v$; an edge (x, y) belongs to $E_{2,h}$ if $(x, y) \in E_{1,h}$ or $x, y \in P \cup N'$ and $(g_2(x), g_2(y)) \in E_h$. Naturally, $\pi_2(x) = \pi(g_2(x))$ for all $x \in N_2$ and $O_2 = (O_1 \setminus P) \cup \{x' \in N' \mid x \in Q_2\}$. The reflexive and transitive closure of \xrightarrow{A} is denoted by \xrightarrow{A}^* and called a *derivation*. For $e \in E_0$ we let O_e such that $d(e) = Q_1 \rightarrow O_e$ and we call the configuration (G_e, O_e, id) an *initial configuration* of A . A *final configuration* is a configuration (M, \emptyset, g) without active nodes. The graph language accepted by the SA-hypergraph automaton A is

$$\mathcal{L}(A) = \left\{ M \in \mathcal{G} \mid \exists e \in E_0: (G_e, O_e, id) \xrightarrow{A}^* (M, \emptyset, g) \right\}.$$

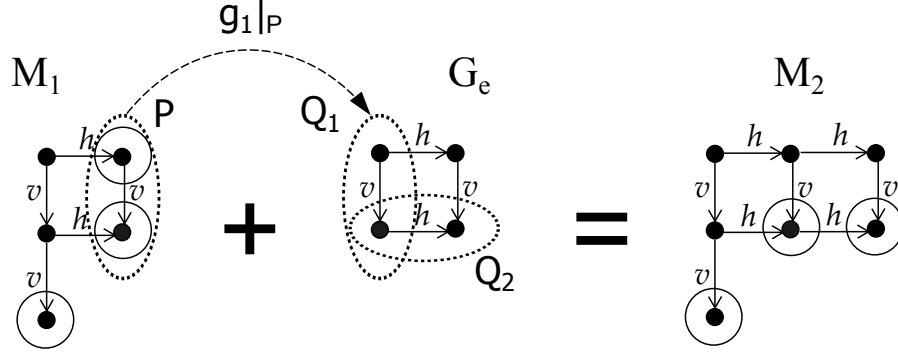


Figure 3.3: A transition $(M_1, O_1, q_1) \rightarrow_A (M_2, O_2, q_2)$ joins together the graphs M_1 and G_e by identifying every node $x \in P$ with the corresponding node $g_1(x) \in Q_1$. The set O_2 of the active nodes of the new configuration M_2 consists of the nodes of the union of the active nodes in $O_1 \setminus P$ with the outgoing active nodes Q_2 of G_e . The active nodes of M_1 and M_2 are represented as circled nodes.

Note that $\mathcal{L}(A)$ contains picture graphs only. The *picture language associated to the graph language* $\mathcal{L}(A)$ is the language $p(\mathcal{L}(A))$.

Remark Since we only talk about picture graphs, we can assume that for every hyperedge $e \in E$ the underlying graph G_e is a subgrid, or e can be removed from the set E .

Example Figure 3.4 shows an example of a self-assembled coloured pattern and an SA-hypergraph automaton that accepts that pattern. Part i) depicts a coloured self-assembled pattern. Parts ii) and iii) together depict the underlying graph of the SA-hypergraph automaton that constructs the same pattern.

The SA-hypergraph automaton for the example in Figure 3.4 is defined as follows. The SA-hypergraph automaton is $A = (N, E, f, d, G, E_0)$, where

- $N = \{x_1, x_2, \dots, x_9, z_1, z_2, \dots, z_7\}$,
- $E = \{e_1, e_2, \dots, e_{16}\}$,

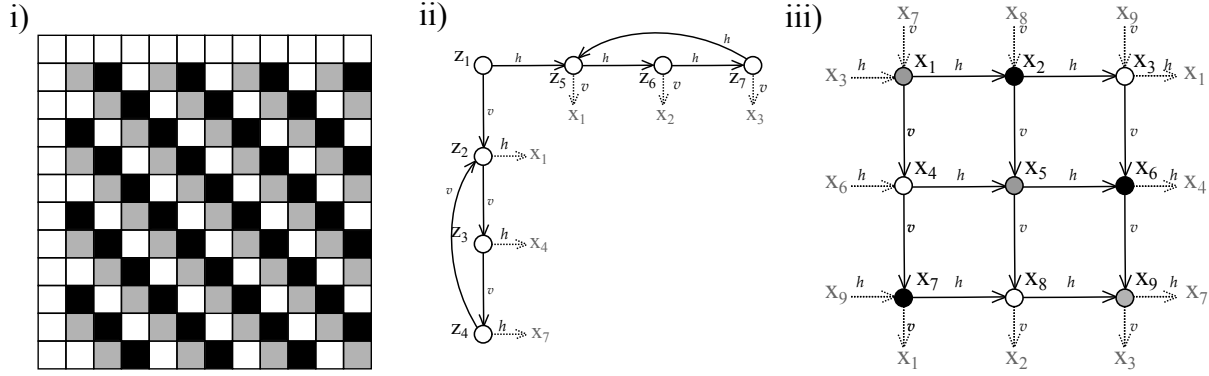


Figure 3.4: Part i) shows an example of coloured self-assembled pattern. Parts ii) and iii) together depict the underlying graph of the SA-hypergraph automaton that constructs the same pattern. Part ii) constructs the white top row and white left column, and part iii) constructs the coloured pattern.

- function f is defined such that

$$\begin{aligned}
 f(e_1) &= \{x_1, x_2, x_4, x_5\}, & f(e_2) &= \{x_2, x_3, x_5, x_6\}, & f(e_3) &= \{x_3, x_1, x_6, x_4\}, \\
 f(e_4) &= \{x_4, x_5, x_7, x_8\}, & f(e_5) &= \{x_5, x_6, x_8, x_9\}, & f(e_6) &= \{x_6, x_4, x_9, x_7\}, \\
 f(e_7) &= \{x_7, x_8, x_1, x_2\}, & f(e_8) &= \{x_8, x_9, x_2, x_3\}, & f(e_9) &= \{x_9, x_7, x_3, x_1\}, \\
 f(e_{10}) &= \{z_1, z_5, z_2, x_1\}, & f(e_{11}) &= \{z_5, z_6, x_1, x_2\}, & f(e_{12}) &= \{z_6, z_7, x_2, x_3\}, \\
 f(e_{13}) &= \{z_7, z_5, x_3, x_1\}, & f(e_{14}) &= \{z_2, x_1, z_3, x_4\}, & f(e_{15}) &= \{z_3, x_4, z_4, x_7\}, \\
 f(e_{16}) &= \{z_4, x_7, z_2, x_1\}.
 \end{aligned}$$

- For each hyperedge in ii), the function d describing the active areas where we can glue new hyperedges is defined as to build a horizontal (vertical) chain of nodes that models the top row (left column) of tiles.

$$\begin{aligned}
 d(e_{10}) &= \{z_1, z_5, z_2\} \rightarrow \{z_5, x_1, z_2\}, \\
 d(e_{11}) &= \{z_5, x_1\} \rightarrow \{z_6, x_1, x_2\}, & d(e_{12}) &= \{z_6, x_2\} \rightarrow \{z_7, x_2, x_3\}, \\
 d(e_{13}) &= \{z_7, x_3\} \rightarrow \{z_5, x_1, x_3\}, & d(e_{14}) &= \{z_2, x_1\} \rightarrow \{z_3, x_1, x_4\}, \\
 d(e_{15}) &= \{z_3, x_4\} \rightarrow \{z_4, x_4, x_7\}, & d(e_{16}) &= \{x_4, z_7\} \rightarrow \{z_2, x_1, x_7\}.
 \end{aligned}$$

The backward edges e.g. (x_3, x_1) , (x_6, x_4) , (x_9, x_7) , and (z_7, z_5) , make it possible to reuse the hyperedges to build a periodic pattern.

For each hyperedge in iii), the function d changes the active input nodes (top-left, bottom-left, and top-right) to the new set of active nodes (top-right, bottom-left, and bottom-right), signifying the change of the places where the new hyperedges can be glued.

$$\begin{aligned}
 d(e_1) &= \{x_1, x_2, x_4\} \rightarrow \{x_2, x_4, x_5\}, & d(e_2) &= \{x_2, x_3, x_5\} \rightarrow \{x_3, x_5, x_6\}, \\
 d(e_3) &= \{x_3, x_1, x_6\} \rightarrow \{x_1, x_6, x_4\}, & d(e_4) &= \{x_4, x_5, x_7\} \rightarrow \{x_5, x_7, x_8\}, \\
 d(e_5) &= \{x_5, x_6, x_8\} \rightarrow \{x_6, x_8, x_9\}, & d(e_6) &= \{x_6, x_4, x_9\} \rightarrow \{x_4, x_9, x_7\}, \\
 d(e_7) &= \{x_7, x_8, x_1\} \rightarrow \{x_8, x_1, x_2\}, & d(e_8) &= \{x_8, x_9, x_2\} \rightarrow \{x_9, x_2, x_3\}, \\
 d(e_9) &= \{x_9, x_7, x_3\} \rightarrow \{x_7, x_3, x_1\}.
 \end{aligned}$$

- Parts ii) and iii) depict the underlying graphs of the white Γ -shaped top and left border of the pattern, and the white-grey-black part of the pattern respectively.
- $E_0 = \{e_{10}\}$

The SA-hypergraph automaton A starts from the top-left white tile, corresponding to $E_0 = \{e_{10}\}$. Afterwards, the automaton continues the construction with the hyperedges in the top row or the left column. The construction of the white-grey-black part starts after the construction of the white top row and left column. Figure 3.5 shows an example of possible transitions of the SA-hypergraph automaton A .

The concept of hypergraph automata was introduced by Janssens and Rozenberg in 1982 [11]. Our definition of SA-hypergraph automata is a variant of the original definition with the following modifications. Firstly, we start from a set of initial graphs whereas the original definition used a single initial graph. For unlabelled graphs both models are capable of accepting the same class of graph languages, as long as one makes an exception for the empty graph. However, for labelled graphs a single initial graph is not sufficient; e. g., if a language L of labelled graphs consists of a graph A where every node is labelled by a and a graph B where every node is labelled by b , then L cannot be generated from the same initial graph since A

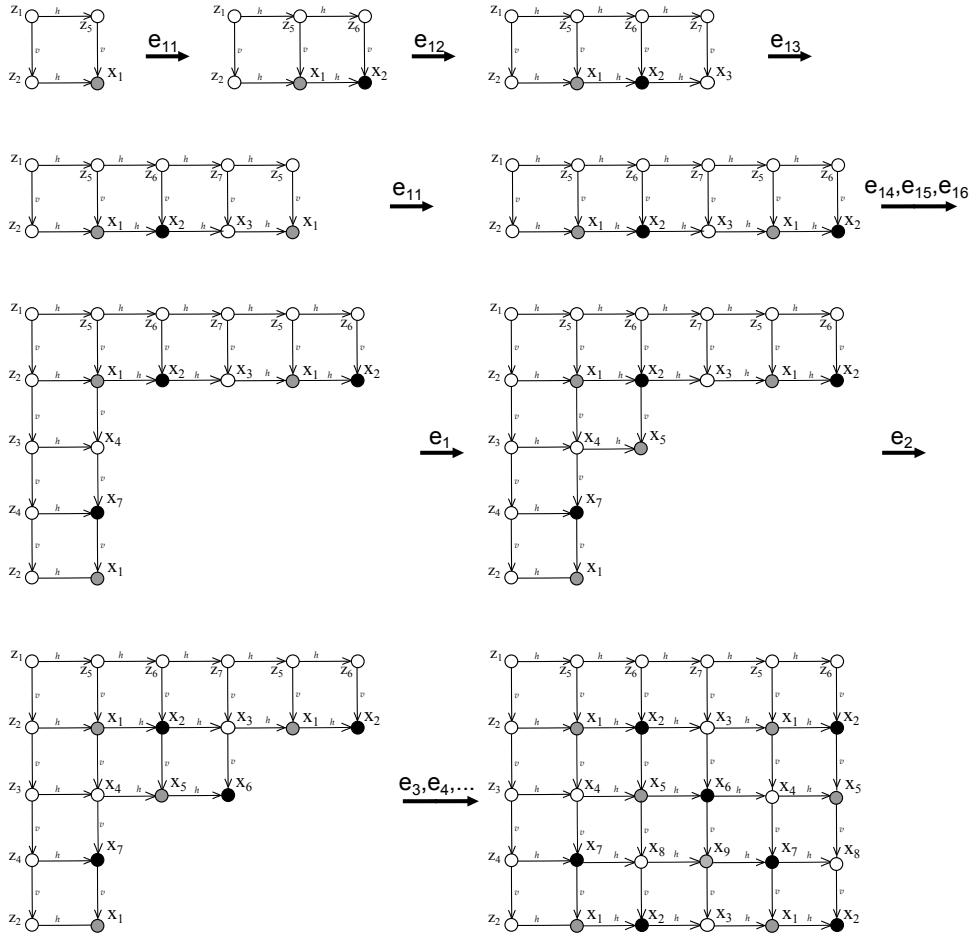


Figure 3.5: In this example, the construction of a picture graph from Figure 3.4 is explained. At each step, one hyperedge or a sequence of hyperedges is glued.

and B do not have a common non-empty isomorphic subgraph. Secondly, we use final configurations in order to accept only some of the graphs that can be generated by rules from the initial graph. In the original definition, for simplicity, final configurations were omitted and every graph which can be generated from the initial graph belonged to the accepted language. Thirdly, it seemed more convenient to us to use the notion of active nodes rather than active intersections.

3.4 Hypergraph Automata for Picture Languages

In this section, we establish a strong connection between (WTS-)recognizable picture languages and picture graph languages that can be accepted by SA-hypergraph automata. We prove that the self-assembly of a Wang Tile System can be simulated by an SA-hypergraph automaton, see Theorem 3.4.1. The main idea is to start the tiling in the top left corner of a tiled picture and then extend the tiled picture downwards and rightwards, just as in Figure 3.2. Our converse result is slightly weaker: the picture language $L = p(\mathcal{L}(A))$, associated to the graph language accepted by an SA-hypergraph automaton A , is WTS-recognizable if A does not contain a *strong loop*, see Theorem 3.4.2. The restriction for A not to contain a strong loop is a natural assumption as strong loops cannot be used in any derivation that accepts a picture graphs.

Theorem 3.4.1 *For any recognizable picture language L there is an SA-hypergraph automaton A such that the picture language associated to the graph language $\mathcal{L}(A)$ is L .*

Proof Let $V = (\Sigma, C', \Theta')$ be a Wang Tile System that recognizes the picture language L , that is $L = \mathcal{L}(V)$. We will slightly modify the WTS V such that it fulfils a certain property as described in the following. We define a WTS $W = (\Sigma, C, \Theta)$ which recognizes L and such that any two copies of a tile $t \in \Theta$ in a tiling of W must have a row- and a column-distance which is a multiple of 3. The modification of V will become of importance later in the proof: We need to ensure that for a 2×2 square of matching tiles t_1, t_2, t_3, t_4 , it is not possible to directly attach another copy of any of t_1, t_2, t_3, t_4 to this square.

We will define a SA-hypergraph automaton $A = (N, E, f, d, G, E_0)$ which simulates the assembly of a tiled picture from $L = \mathcal{L}(W)$ as described in Fig. 3.2 ii). Let N be a set of nodes such that $|N| = |\Theta|$ and let $\vartheta: N \rightarrow \Theta$ be a bijection. For each node $x \in N$ there is a *corresponding tile* $\vartheta(x)$ and vice versa. Let N_T, N_R, N_B, N_L be the set of nodes which correspond to tiles on the top, right, bottom, left border of a tiled picture, respectively:

$$\begin{aligned} N_T &= \{x \in N \mid \sigma_N(\vartheta(x)) = \#\}, & N_R &= \{x \in N \mid \sigma_E(\vartheta(x)) = \#\}, \\ N_B &= \{x \in N \mid \sigma_S(\vartheta(x)) = \#\}, & N_L &= \{x \in N \mid \sigma_W(\vartheta(x)) = \#\}. \end{aligned}$$

Let $G = (N, E_v \cup E_h, \pi)$ be the underlying graph of A . The label function π is naturally defined as $\pi(x) = \lambda(\vartheta(x))$ for $x \in N$. For all nodes $x, y \in N$ there is an edge $(x, y) \in E_h$ if and only if $\sigma_E(\vartheta(x)) = \sigma_W(\vartheta(y)) \neq \#$ and either $x, y \in N \setminus (N_T \cup N_B)$ or $x, y \in N_T$ or $x, y \in N_B$; there is an edge $(x, y) \in E_v$ if and only if $\sigma_S(\vartheta(x)) = \sigma_N(\vartheta(y)) \neq \#$ and either $x, y \in N \setminus (N_L \cup N_R)$ or $x, y \in N_L$ or $x, y \in N_R$. This means if the east edge of a tile t can attach to the west edge of tile s , then their corresponding nodes $x = \vartheta^{-1}(t)$ and $y = \vartheta^{-1}(s)$ are connected by an h -edge $(x, y) \in E_h$. Analogously, if the south edge of a tile t can attach to the north edge of tile s , then their corresponding nodes $x = \vartheta^{-1}(t)$ and $y = \vartheta^{-1}(s)$ are connected by an v -edge $(x, y) \in E_v$.

If $N_T \cap N_B \neq \emptyset$ or $N_R \cap N_L \neq \emptyset$, the language $\mathcal{L}(W)$ possibly contains pictures p with $h(p) = 1$ or $w(p) = 1$, respectively, which can be seen as one-dimensional pictures. These pictures have to be treated separately. For now we assume that $N_T \cap N_B = N_R \cap N_L = \emptyset$.

The hyperedges E and the transition function d define the possible transitions of A . In every transition we add exactly one node to the graph of a configuration of A . Our naming convention is that x is the node which is attached in the derivation step and y, y_1, y_2, y_3 are incoming active nodes of the hyperedge. Every graph containing only one node which corresponds to a tile in the top left corner is an initial graph. In order to construct a picture graph which represents a picture in $\mathcal{L}(W)$ we introduce three types of transitions, see Figure 3.6. The transitions of type I generate the top row of the graph and transitions of type II generate the left column of the graph; both transition types keep every generated node active. Transitions of type III generate the rest of the graph: A node is attached if it has a matching east neighbour (y_1), a matching north neighbour (y_3), and these two nodes are connected by another node (y_2); unless we reach the right or bottom border of the graph the nodes x, y_1 , and y_3 are active after using the transition.

Formally, we define the set of hyperedges E , the set of initial edges E_0 , the function f , and the transition function d as following:

Initial hyperedges: For each $x \in N_T \cap N_L$, corresponding to a tile in the top left corner, we define a hyperedge $e_x \in E_0 \subseteq E$ with associated nodes $f(e_x) = \{x\}$ and the transition function $d(e_x) = \emptyset \rightarrow \{x\}$.

Type I: For all nodes $x, y \in N_T$, in the top row, such that $(x, y) \in E_h$, we define a hyperedge $e_{x,y} \in E$ with associated nodes $f(e_{x,y}) = \{x, y\}$ and the derivation function $d(e_{x,y}) = \{y\} \rightarrow \{x, y\}$.

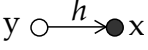
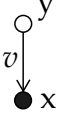
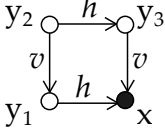
Type	I	II	III
Hyperedges			

Figure 3.6: The hyperedges in the SA-hypergraph automaton A induce three different types of graphs. White nodes represent incoming active nodes of the hyperedges.

Type II: For all nodes $x, y \in N_L$, in the left column, such that $(x, y) \in E_v$, we define a hyperedge $e_{x,y} \in E$ with associated nodes $f(e_{x,y}) = \{x, y\}$ and the derivation function $d(e_{x,y}) = \{y\} \rightarrow \{x, y\}$.

Type III: For all nodes $x \in N \setminus (N_T \cup N_L)$ and $y_1, y_2, y_3 \in N$ such that $(y_2, y_1), (y_3, x) \in E_v$ and $(y_2, y_3), (y_1, x) \in E_h$, we define a hyperedge $e_{x,y_1,y_2,y_3} \in E$ with associated nodes $f(e_{x,y_1,y_2,y_3}) = \{x, y_1, y_2, y_3\}$ and the derivation function

1. $d(e_{x,y_1,y_2,y_3}) = \{y_1, y_2, y_3\} \rightarrow \emptyset$ if $x \in N_B \cap N_R$, (bottom right corner)
2. $d(e_{x,y_1,y_2,y_3}) = \{y_1, y_2, y_3\} \rightarrow \{x, y_3\}$ if $x \in N_B \setminus N_R$, (bottom row)
3. $d(e_{x,y_1,y_2,y_3}) = \{y_1, y_2, y_3\} \rightarrow \{x, y_1\}$ if $x \in N_R \setminus N_B$, (right column)
4. $d(e_{x,y_1,y_2,y_3}) = \{y_1, y_2, y_3\} \rightarrow \{x, y_1, y_3\}$ otherwise.

Consider the graph G_e which is induced by the hyperedge $e \in E$. Depending on the type of the hyperedge e , the graph G_e contains at least the edges shown in Figure 3.6. However, by the modification of the Wang tile system V above, we ensured that the graph G_e contains exactly those edges shown in Figure 3.6. Suppose one of the graphs G_e would contain an edge (x', y') which is not shown in Figure 3.6, then the tile corresponding to y' could occur in two positions which are less than three rows and columns apart — a property that was excluded by the modification.

We will show that $p(\mathcal{L}(A)) = L$. Firstly, consider an array \bar{p} of tiles from Θ which is the Wang-tiled version of the picture $p \in \mathcal{L}(W)$. We will show that the SA-hypergraph automaton A accepts a picture graph M such that $p(M) = p$. We assume M to be embedded in \mathbb{Z}^2 such

that the nodes cover the axis-parallel rectangle spanned by the points $(1, 1)$ and $(h(p), w(p))$, every v -edge points downwards, and every h -edge points rightwards; recall that our coordinates represent the rows and columns of a matrix. The derivation leading to the final configuration (M, \emptyset, g) simulates the assembly of tiles which form \bar{p} as shown in Figure 3.2. The north and west edges of the tile $t_{TL} = \bar{p}_{(1,1)}$ in the top left corner of \bar{p} are labelled by $\#$, and therefore, the node $x_{TL} = \vartheta^{-1}(t_{TL})$ corresponding to t_{TL} forms an initial graph M_0 . The adjacent edges of two neighbouring tiles s, t in \bar{p} are labelled by the same colour. Suppose s is the west neighbour of t , then $\sigma_E(s) = \sigma_W(t) \neq \#$ and both tiles belong to the same row, implying that $\sigma_N(s) = \# \iff \sigma_N(t) = \#$ and $\sigma_S(s) = \# \iff \sigma_S(t) = \#$. Therefore, their corresponding nodes in G are connected by an h -edge $(\vartheta^{-1}(s), \vartheta^{-1}(t)) \in E_h$. Analogously, if s is the north neighbour of t , then $(\vartheta^{-1}(s), \vartheta^{-1}(t)) \in E_v$. Next, we see that the hyperedges of type I and type II can be used in order to create the top row and left column of the graph M , respectively. Furthermore, the hyperedges of type III can be used in order to create all the remaining nodes of M . We conclude that $(M_0, \{x_{TL}\}, id) \xrightarrow[A]{*} (M, O, g)$ is a derivation in A and we will prove that (M, O, g) has to be a final configuration with $O = \emptyset$. Observe, that hyperedges of types I and II leave all the nodes active while hyperedges of type III deactivate at least the top left node in the hyperedge. Thus, all nodes except for those in the bottom row and in the right column will be deactivated in the configuration (M, O, g) . Furthermore, in order to create the bottom row and right column hyperedges of type III.2 and III.3 are used, respectively, and one rule of type III.1 is used in order to create the bottom-right node of M . It is easy to see that the derivation function is designed such that all nodes will be deactivated in the configuration (M, O, g) and, therefore, A accepts M .

Now, let $M = (N_M, E_{v,M} \cup E_{h,M}, \pi_M) \in L(A)$ be a graph which is generated by A . Let G be accepted by the derivation

$$(M_0, O_0, g_0) \xrightarrow[A]{*} (M_1, O_1, g_1) \xrightarrow[A]{*} \cdots \xrightarrow[A]{*} (M_k, O_k, g_k)$$

where $(M_0, O_0, g_0) = (G_{e_0}, O_{e_0}, id)$ is an initial configuration with $e_0 \in E_0$ and $(M_k, O_k, g_k) = (M, \emptyset, g)$ is a final configuration. Let N_i be the node set of the graph M_i . Note that for any $0 \leq i \leq k$ the function g_i is the restriction of g by N_i , that is $g_i = g|_{N_i}$. In order to avoid

confusion, nodes in the graph M are consistently denoted by x, y and nodes in the graph G are consistently denoted by x', y' ; the nodes may have subscripts.

Let the nodes in the graphs M_0, \dots, M_k be embedded in \mathbb{Z}^2 such that all h -edges point rightwards and all v -edges point downwards; just like we did above. The creation of graph $M = M_k$ starts with the initial graph M_0 which contains only one node $x_{TL} \in N_T \cap N_L$. Let x_{TL} lie on position $(1, 1)$ in all of the graphs M_0, \dots, M_k . The graph M_0 can be extended rightwards by using hyperedges of type I and downwards by hyperedges of type II. Since none of the hyperedges attach a new node upwards or leftwards of an existing node in M_{i-1} in order to obtain M_i , the node x_{TL} lies in the top row and in the left column of M_i . By the definition of type I and II hyperedges, for every node y in the top row (resp., left column) of M we have $g(y) \in N_T$ (resp., $g(y) \in N_L$). By using hyperedges of type III the area spanned by the top row and left column can be filled with nodes. It is easy to see that for all graphs M_0, \dots, M_k we have that if a node lies on position (i, j) , then for all $(i', j') \in [i] \times [j]$ a node lies on position (i', j') . Furthermore, if $i' < i$, then the node on position (i', j') has an outgoing v -edge, and if $j' < j$, then the node on position (i', j') has an outgoing h -edge. In other words, in the axis-parallel rectangle spanned by the points $(1, 1)$ and (i, j) all nodes are connected by edges with all direct neighbours (nodes which have an Euclidean distance of 1).

In the final configuration (M_k, \emptyset, g) there are no active nodes. Thus, the last node which is added to the graph M_{k-1} in order to obtain M_k is a node x_{BR} such that $g(x_{BR}) \in N_B \cap N_R$, as all other derivation rules will leave some nodes active. Next, let us consider the nodes which belong to the same row and column as x_{BR} does. Note that if two nodes x and y in M are connected by an edge, then the corresponding nodes $g(x)$ and $g(y)$ in G are connected by an edge, too; more precisely, if $(x, y) \in E_{v,M}$, then $(g(x), g(y)) \in E_v$, and if $(x, y) \in E_{h,M}$, then $(g(x), g(y)) \in E_h$. Since a node in N_B (resp., N_R) is connected only by h -edges (resp., v -edges) in G to other nodes from N_B (resp., N_R), we see that for every node y in the row of x_{BR} (resp., column of x_{BR}) we have $g(y) \in N_B$ (resp., $g(y) \in N_R$). A node $y' \in N_B$ (resp., $y' \in N_R$) does not have any outgoing v -edges (resp., h -edges) as the south edge (resp., east edge) of $\vartheta(y')$ is labelled by $\#$. We conclude that x_{BR} sits in the bottom row and right column of the graph M and, by the observations made above, this implies that M is a picture graph.

We claim that the picture $p(M)$ which corresponds to the graph M can be generated by the

assembly \bar{p} given by the embedding of nodes in M and the function $\vartheta \circ g$. Clearly, for every node y on position (i, j) in M we have that $p(M)_{i,j} = \pi_M(y) = \lambda(\vartheta(g(y)))$, therefore, the pictures p and $\lambda(\bar{p})$ coincide. Next, we prove that \bar{p} is a tiled picture in the Wang tile system W . Recall, that all nodes on the top, right, bottom, and left border of M correspond to tiles in M_T, M_R, M_B , and M_L , respectively, and therefore, \bar{p} is well-bordered. Let t_x and t_y be two neighbouring tiles in \bar{p} which lie on positions (i, j) and $(i, j + 1)$, respectively. Let x and y be the nodes in M which lie on the positions (i, j) and $(i, j + 1)$, respectively. Note that $t_x = \vartheta(g(x))$ and $t_y = \vartheta(g(y))$. Since M is a picture graph, $(x, y) \in E_{h,M}$ and $(g(x), g(y)) \in E_h$. The edge set E_h was built to ensure that $\sigma_E(t_x) = \sigma_W(t_y)$. We conclude that all adjacent east-west edges in \bar{p} have matching colours. By symmetric arguments, we also conclude that all adjacent north-south edges in \bar{p} have matching colours. Therefore, \bar{p} is a tiled picture in W and $p \in L$.

Finally, let us consider the case when $N_T \cap N_B \neq \emptyset$. We can add a component to the SA-hypergraph automaton which works similar to a non-deterministic finite automaton and where every hyperedge induces an graph of type I in Figure 3.6. The initial graphs are given by all nodes from $N_T \cap N_B \cap N_L$. For all nodes $x, y \in N_T \cap N_B$ with $(y, x) \in E_h$ we define a hyperedge $e_{x,y}$ such that $f(e_{x,y}) = \{x, y\}$. The derivation function is given as $d(e_{x,y}) = \{y\} \rightarrow \{x\}$ if $x \notin N_R$, and $d(e_{x,y}) = \{y\} \rightarrow \emptyset$ otherwise. Obviously, this attachment to the hypergraph A accepts all graphs which correspond to pictures $p \in L$ with $h(p) = 1$. The case when $N_L \cap N_R \neq \emptyset$ can be covered analogously.

Next, we prove that a picture language $L = p(\mathcal{L}(A))$, associated to the graph language $\mathcal{L}(A)$, is WTS-recognizable if A does not contain a strong loop.

Let A be an SA-hypergraph automaton. A series of hyperedges $s = \langle e_0, e_1, \dots, e_n \rangle$ from A is a (*derivation*) *loop* if $e_0 = e_n$ and $Q_{2,i} \cap Q_{1,i+1} \neq \emptyset$ where $d(e_i) = Q_{1,i} \rightarrow Q_{2,i}$ for $0 \leq i < n$. Loops in an SA-hypergraph automaton are a prerequisite for using a hyperedge several times in one derivation. Therefore, an SA-hypergraph automaton without any loops can only accept a finite graph language. Let $G_i = G_{e_i}$ be the graph induced by e_i , let x be a node in $G_0 = G_n$, and let $O_i = Q_{2,i} \cap Q_{1,i+1}$ be the set overlapping incoming/outgoing active nodes of G_i and G_{i+1} . There is a path in the underlying graph of A from x to x which only visits the subgraphs G_0, \dots, G_n , in the given order, and passes through at least one node of each O_i (the path may use incoming and outgoing edges). The loop s is a *strong loop* if, on this path, the number of

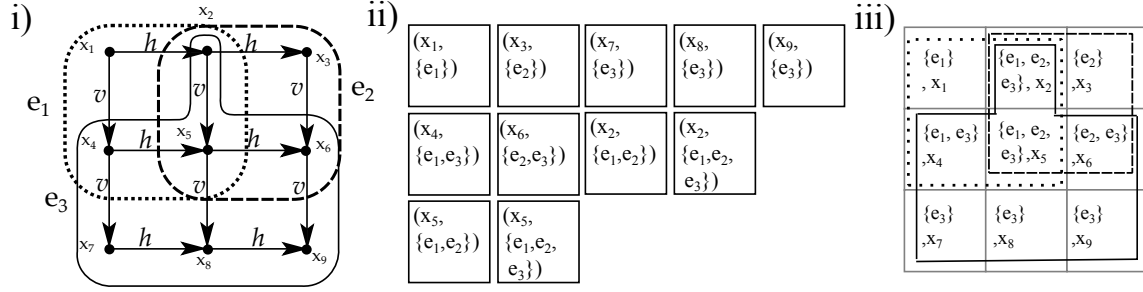


Figure 3.7: Let $A = (N, E, f, d, G, E_0)$ be an SA-hypergraph automaton where N , E , f , and G are defined in part i). function d is defined such that $d(e_1) = \{x_1\} \rightarrow \{x_2, x_4, x_5\}$, $d(e_2) = \{x_2, x_5\} \rightarrow \{x_2, x_5, x_6\}$ and $d(e_3) = \{x_2, x_4, x_5, x_6\} \rightarrow \{\}$. SA-hypergraph automaton starts from e_1 . Part ii) shows the set of all the possible tile candidates. On each tile related node and the set of ψ are written. The tiling on part iii) is the result of overlapping of three hyperedges e_1, e_2 and e_3 .

incoming horizontal edges equals the number of outgoing horizontal edges and the number of incoming vertical edges equals the number of outgoing vertical edges. In other words, when starting from a configuration M and successively gluing the hyperedges from s to M , then the subgraph added by the hyperedge e_0 and the subgraph added by the hyperedge e_n fully overlap when naturally embedded in \mathbb{Z}^2 . Note that, by Remark 3.3, all graphs G_i are subgrids which implies that the choice of the path from x to x does not matter in this definition.

Theorem 3.4.2 *Let A be an SA-hypergraph automaton without any strong loops. The picture language $L = p(\mathcal{L}(A))$, associated to the graph language $\mathcal{L}(A)$, is WTS-recognizable (Wang tile system recognizable).*

Proof Let $A = (N, E, f, d, G, E_0)$ and let $G = (N, E_v \cup E_h, \pi)$. We may assume that $e \in E_0$ if and only if $d(e) = \emptyset \rightarrow O_e$. Therefore, none of the initial hyperedges can be used in a transition. This assumption is justified by the fact that we can duplicate all hyperedges in E_0 such that one copy can be used in a transition but does not belong to E_0 and the other copy which belongs to E_0 cannot be used in a transition. Furthermore, any hyperedge without incoming active nodes which does not belong to E_0 is useless and can be removed from E .

For a node $x \in N$ we define the list of *related hyperedges* to x , $H_x = \{e \in E \mid x \in f(e)\}$. Let x be a node and $\psi \subseteq H_x$. We call a hyperedge $g \in \psi$ a *generator* of (x, ψ) if $x \notin Q_1$ with $d(g) = Q_1 \rightarrow Q_2$. Note that if $g \in E_0$, then g must be a generator. We call a hyperedge $c \in \psi$

a *consumer* of (x, ψ) if $x \notin Q_2$ with $d(c) = Q_1 \rightarrow Q_2$. The pair (x, ψ) is a *tile candidate* if ψ contains exactly one generator $g_{(x, \psi)}$ and exactly one consumer $c_{(x, \psi)}$; furthermore, if $g_{(x, \psi)} = c_{(x, \psi)}$, we require that $\psi = \{g_{(x, \psi)}\}$. Note that if $g_{(x, \psi)} \neq c_{(x, \psi)}$, then for all $e \in \psi$ with $d(e) = Q_1 \rightarrow Q_2$, we have that $x \in Q_1$ unless e is the generator and $x \in Q_2$ unless e is the consumer. The tile candidate (x, ψ) describes the attachment of a copy of the node x to the output graph by the generator; afterwards, x is used as active node by all hyperedges in $\psi \setminus \{g_{(x, \psi)}, c_{(x, \psi)}\}$; finally, x is deactivated by the consumer. Let G_ψ be the node-induced subgraph of G by $\bigcup_{e \in \psi} f(e)$. If G_ψ is not a subgrid (a subgraph of some picture graph), we remove (x, ψ) from the set of tile candidates. Let Ψ denote the set of all remaining tile candidates.

The Wang tile system $W = (\Sigma, C, \Theta)$ which recognizes L is constructed based on the list Ψ . In order to recognize the picture language associated to $\mathcal{L}(A)$, we have to define the attachments of tile candidates. We use unordered pairs $\{(x, \psi), (y, \varphi)\} \in \Psi^2$ of tile candidates for the colours on the edges. For a tile candidate $(x, \psi) \in \Psi$ we define the set of labelled Wang tiles

$$\Theta_{(x, \psi)} = \mathcal{S}_{N, (x, \psi)} \times \mathcal{S}_{E, (x, \psi)} \times \mathcal{S}_{S, (x, \psi)} \times \mathcal{S}_{W, (x, \psi)} \times \{l_x\}$$

where l_x is the label $\pi(x)$ and $\mathcal{S}_{N, (x, \psi)}$, $\mathcal{S}_{E, (x, \psi)}$, $\mathcal{S}_{S, (x, \psi)}$, $\mathcal{S}_{W, (x, \psi)}$ are sets of colours which are defined below. The set of all tiles is the union $\Theta = \bigcup_{(x, \psi) \in \Psi} \Theta_{(x, \psi)}$.

For $(x, \psi), (y, \varphi) \in \Psi$, we let $\{(x, \psi), (y, \varphi)\} \in \mathcal{S}_{E, (x, \psi)}$ and $\{(x, \psi), (y, \varphi)\} \in \mathcal{S}_{W, (y, \varphi)}$ if and only if

1. $(x, y) \in E_h$,
2. $H_x \cap \varphi \subseteq \psi$,
3. $\psi \cap H_y \subseteq \varphi$, and
4. $g_{(x, \psi)} = g_{(y, \varphi)}$ or $y \in Q_1$ for $d(g_{(x, \psi)}) = Q_1 \rightarrow Q_2$ or $x \in Q'_1$ for $d(g_{(y, \varphi)}) = Q'_1 \rightarrow Q'_2$.

For $(x, \psi), (y, \varphi) \in \Psi$, we let $\{(x, \psi), (y, \varphi)\} \in \mathcal{S}_{S, (x, \psi)}$ and $\{(x, \psi), (y, \varphi)\} \in \mathcal{S}_{N, (y, \varphi)}$ if and only if $(x, y) \in E_v$ and conditions 2 to 4 are satisfied. For $(x, \psi) \in \Psi$, we let $\mathcal{S}_{E, (x, \psi)} = \{\#\}$ if x does not have an outgoing horizontal edges in the graph G_ψ . By symmetric condition we let $\mathcal{S}_{N, (x, \psi)} = \{\#\}$, $\mathcal{S}_{S, (x, \psi)} = \{\#\}$, or $\mathcal{S}_{W, (x, \psi)} = \{\#\}$.

Now, consider an $m \times n$ -picture graph $M = (N_M, E_{v,M} \cup E_{h,M}, \pi_M) \in \mathcal{L}(A)$. We will show that there is a tiled version \bar{p} of picture $p = p(M)$ which uses tiles from Θ and, therefore, p is recognized by W . Let G be accepted by the derivation

$$(M_0, O_0, g_0) \xrightarrow{A} (M_1, O_1, g_1) \xrightarrow{A} \cdots \xrightarrow{A} (M_k, O_k, g_k)$$

where $(M_0, O_0, g_0) = (G_{e_0}, O_{e_0}, id)$ is an initial configuration (that is $e_0 \in E_0$) and $(M_k, O_k, g_k) = (M, \emptyset, g)$ is a final configuration. Let e_i be the hyperedge and $P_i \subseteq O_{i-1}$ be the active nodes which are used in the transition $(M_{i-1}, O_{i-1}, g_{i-1}) \xrightarrow{A} (M_i, O_i, g_i)$. Let $d(e_i) = Q_{1,i} \rightarrow Q_{2,i}$ for $1 \leq i \leq k$. Recall that, by definition, M_{i-1} is a full subgraph of M_i and, by induction, every graph M_i is a full subgraph of M . Being an $m \times n$ -picture graph, the nodes in M can be naturally embedded in $[m] \times [n]$ by the function f_M .

Consider one node $x' \in N_M$ and its original $x = g(x')$ in G . We assign to x' a list of hyperedges $\psi \subseteq E$ such that $e_i \in \psi$ if $x' \in P_i$ or x' belongs to M_i but not M_{i-1} . We intend to use a tile from $\Theta_{(x,\psi)}$ for the pixel $\bar{p}_{f_M(x')}$ representing x' in the tiled picture \bar{p} . Observe that ψ contains a consumer as x' is not active in the final configuration and ψ cannot contain two consumers because a node can only be deactivated once. In addition, the hyperedge e_i such that x' belongs to M_i but not M_{i-1} is the single generator in ψ . Since G_ψ is isomorphic to a subgraph of M , we conclude that (x, ψ) is indeed a tile candidate. If x' does not have an outgoing horizontal edge, then the node x in the graph G_ψ cannot have an outgoing horizontal edge either and, therefore, $\mathcal{S}_{E,(x,\psi)} = \{\#\}$. Symmetric arguments apply if x does not have an incoming horizontal, outgoing vertical, or incoming vertical edge.

Next, consider two nodes $x', y' \in N_M$ which are connected by an edge and, by symmetry, assume (x', y') is a horizontal edge. Let $x = g(x')$, $y = g(y')$ be their originals and let ψ, φ be the set of hyperedges associated to x', y' , respectively. We will show that $\{(x, \psi), (y, \varphi)\}$ is a colour in $\mathcal{S}_{E,(x,\psi)}$ as well as in $\mathcal{S}_{W,(y,\varphi)}$. Thus, we can choose tiles from $\Theta_{(x,\psi)}$ and $\Theta_{(y,\varphi)}$ for the positions $f_M(x')$ and $f_M(y')$ in \bar{p} , respectively. Clearly, the choice of the tiles also depends on the other neighbours of x' and y' . We have to show that conditions 1 to 4, above, are satisfied. The first condition is satisfied by assumption. By contradiction, suppose the second condition is not satisfied. There is $e_i \in H_x \cap \varphi \setminus \psi$; thus, in the i -th step of the derivation we use the

hyperedge e_i that presupposes or generates an edge (x'', y') in M where $g(x'') = x$ but $x'' \neq x'$. This would imply that y has two incoming horizontal edges whence M is not a picture graph. The third condition is satisfied by symmetric arguments. The edge (x', y') in M can only be created in step i where x' or y' is added to the graph M_{i-1} . Thus, x' and y' either have the same generator in (x, ψ) and (y, φ) , or x' is in the active nodes when y' is generated, or y' is in the active nodes when x' is generated. In all cases condition 4 is satisfied.

We conclude that a tiled picture \bar{p} such that $p = p(M)$ and $M \in \mathcal{L}(A)$ can be generated by using tiles from Θ and, therefore, $p(M) \in \mathcal{L}(W)$.

Consider a picture $p \in \mathcal{L}(W)$ and let \bar{p} be the tiled version of p , using tiles from $\Theta = \bigcup_{(x, \psi) \in \Psi} \Theta_{(x, \psi)}$.

We start by introducing the concept of masks which can be seen as connected subpictures of the tiled picture \bar{p} that represent the nodes in one hyperedge. A *mask* m is a $h(\bar{p}) \times w(\bar{p})$ matrix of tiles from $\Theta \cup \{\text{empty}\}$, such that either $m_{(i, j)} = \text{empty}$ or $m_{(i, j)} = \bar{p}_{(i, j)}$ for all $(i, j) \in [h(\bar{p})] \times [w(\bar{p})]$. In addition, we require that the non-empty entries in m are connected; that is, for every pair of tiles $m_{(i', j')}, m_{(i, j)} \in \Theta$ there exists a sequence $r = \langle r_0, r_1, \dots, r_n \rangle$ of tiles in m such that $r_0 = m_{(i, j)}$, $r_n = m_{(i', j')}$, $r_k \in \Theta$, and r_k, r_{k+1} must be adjacent for all $0 \leq k < n$.

Let $e \in E$ be an hyperedge and let $G_e = (N_e, E_{e, v} \cup E_{e, h}, \pi_e)$ be the graph induced by this hyperedge. By Remark 3.3, we assume that G_e is a subgrid. We say G_e is mapped to a mask m if there is a injective function $h: N_e \rightarrow [h(\bar{p})] \times [w(\bar{p})]$ which satisfies: $m_{(i, j)}$ belongs to Θ if and only if (i, j) is in the domain of h ; for all nodes $x, y \in N_e$ there is an edge $(x, y) \in E_{e, h}$ (resp., $(x, y) \in E_{e, v}$) if and only if $h(x)$ is in north (resp., west) neighbour of $h(y)$. Whenever we use this mapping, we will ensure that for all $x \in G_e$ the tile $\bar{p}_{h(x)}$ belongs to $\Theta_{(x, \psi)}$ for some $\psi \subseteq E$.

Consider a tile $t \in \bar{p}_{(i, j)} \in \Theta_{(x, \psi)}$ and a hyperedge $e \in \psi$. We define the mask $m^{[(i, j), x, e]}$ such that the graph G_e can be mapped by function h to $m^{[(i, j), x, e]}$ and $h(x) = (i, j)$. We say that e is the hyperedge related to the mask $m^{[(i, j), x, e]}$. Let $t' = \bar{p}_{(i', j')} \in \Theta_{(y, \varphi)}$ be a tile that is adjacent to t and let $e \in \psi$. For simplicity we only consider the case when t' is the east neighbour of t ; i.e., $(i', j') = (i, j + 1)$. We will show that if (i', j') is non-empty in $m^{[(i, j), x, e]}$, then $e \in \varphi$. Since t' is the east neighbour of t conditions 1 to 4, above, apply. As (i', j') is non-empty in $m^{[(i, j), x, e]}$, there exists a horizontal edge (x, z) in G_e . Furthermore, from conditions 1 and 4 it follows that

(x, y) is a horizontal edge in the graph G_g induced by the generator $g = g_{(x, \varphi)}$. As both graphs G_e and G_g are subgraphs of the subgrid G_ψ , we see that the edges (x, y) and (x, z) coincide, thus, $y = z$. We conclude $y \in G_e$ and $e \in H_y$. By condition 3, $e \in \varphi$. Because the hyperedge e induces a connected graph, we can infer that for all non-empty $m_{(i', j''), e}^{[(i, j), x, e]} \in \Theta_{(z, \chi)}$, we find $e \in \chi$. Note that this also implies that $m_{(i, j), x, e}^{[(i, j), x, e]} = m_{(i', j''), y, e}^{[(i', j''), y, e]} = m_{(i'', j''), z, e}^{[(i'', j''), z, e]}$.

We define the set of masks $\mu = \{m_{(i, j), x, e}^{[(i, j), x, e]} | \bar{p}_{(i, j)} \in \Theta_{(x, \psi)}, e \in \psi\}$ which are induced by hyperedges in the above manner. Intuitively, every mask in μ represents one transition in the derivation of a picture graph M which represents the picture $p = p(M)$. In order to use a transition defined by a mask, we need to guarantee that all of its input areas exist and are active. We will order the set μ accordingly. Let us define the relation $R \subseteq \mu \times \mu$ such that $(m, n) \in R$ if the transition represented by m has to be used before the transition represented by n . Let m and n be two distinct masks in μ . The pair (m, n) is in R if there exists (i, j) such that $m_{(i, j)} = n_{(i, j)} \in \Theta_{(x, \psi)}$, and $m = m_{(i, j), x, g}^{[(i, j), x, g]}$ where $g = g_{(x, \psi)}$ or $n = m_{(i, j), x, c}^{[(i, j), x, c]}$ where $c = c_{(x, \psi)}$. The pair (μ, R) can be seen as directed graph G_μ . First, we show that the graph G_μ does not contain any loops, afterwards, a topological sort of this graph is used to order the transitions represented by the masks.

When two masks overlap on a tile (have a common non-empty entry), regarding the construction of tile candidates, we know that the related hyperedge of exactly one of these masks is the generator of the input area of the other hyperedges. Hence, these masks are connected in the graph G_μ . By contradiction, assume that $\langle n_0, n_1, \dots, n_l \rangle$ is a non-trivial loop in G_μ (i.e., $(n_i, n_{i+1}) \in R$ for every $0 \leq i < l - 1$ and $(n_l, n_0) \in R$). However, the sequence of related hyperedges to this sequence of mask is a strong loop in the SA-hypergraph automaton A which was excluded by assumption. Moreover, since two tiles with different generators cannot connect without satisfying conditions 4, the graph G_μ must be connected. Therefore, graph G_μ can be topologically sorted. Sorting of the hyperedges guaranteed that the active input nodes of one hyperedge are generated before the gluing of the hyperedge.

By contradiction, assume that graph G_μ has two distinct nodes m_1 and m_2 without any input edges. Let m_3 be the first node in the topological order such that paths $m_1 \rightarrow^* m_3$ and $m_2 \rightarrow^* m_3$ exist in G_μ . As m_3 is chosen minimal, these paths do not share any node other than m_3 . Recall that all incoming active nodes of a hyperedge are connected. Considering that two

nodes cannot connect to each other unless they are in the same hyperedge or they have glued to each other, we have a contradiction as m_3 cannot be the first common node on both paths. We conclude that graph G_μ has only one node without input.

Now, let m_0, m_1, \dots, m_k be the topological sort of μ by the relation \mathcal{R} . We define $m + n = o$ such that $o_{(i,j)} = \text{empty}$ if $m_{(i,j)} = n_{(i,j)} = \text{empty}$; otherwise, $o_{(i,j)} = \bar{p}_{(i,j)}$. We will show that a graph M_k can be generated by a derivation

$$(M_0, O_0, g_0) \xrightarrow{A} (M_1, O_1, g_1) \xrightarrow{A} \cdots \xrightarrow{A} (M_k, O_k, g_k)$$

such that the graph M_i can be mapped to the mask $\sum_{j=0}^i m_j$; this implies that m_k can be mapped to $\bar{p} = \sum_{j=0}^k m_j$. Let e_i be the hyperedge related to the mask m_i . The graph $M_0 = G_{e_0}$ is an initial graph because m_0 has no incoming edges in \mathcal{G}_μ and, therefore, the derivation function of e_0 is $d(e_0) = \emptyset \rightarrow Q_2$; thus, (M_0, O_0, g_0) where $O_0 = Q_2$ and $g_0 = id$ is an initial configuration. In derivation step $(M_{i-1}, O_{i-1}, g_{i-1}) \xrightarrow{A} (M_i, O_i, g_i)$ we use the hyperedge e_i . By induction, we can assume that M_{i-1} can be mapped to $\sum_{j=0}^{i-1} m_j$ by a function h_{i-1} . There is only one way to glue the hyperedge e_i to M_{i-1} such that resulting graph M_i can be mapped to $\sum_{j=0}^i m_j$. We have to prove that all incoming active nodes of G_{e_i} exist and are active in M_i . Let x be an incoming active node which is represented by the tile $\bar{p}_{(a,b)} \in \Theta_{(x,\psi)}$. The definition of R ensures that the mask representing the generator of (x, ψ) in (a, b) has already been used and that the mask representing the consumer of (x, ψ) in (a, b) has not yet been used. Finally, every tile candidate has a consumer which means that there are no active nodes in the final configuration (M_k, O_k, g_k) . As result, the picture p , generated by the suggested tiling system, is in $p(\mathcal{L}(A))$.

3.5 Conclusion

We introduced SA hypergraph automata, a language/automata theoretic model for patterned self-assembly systems. SA hypergraph automata accept all recognizable picture languages but, unlike other models, (e.g. Wang Tile Automata) SA-hypergraph automata do not rely on an *a priori* given scanning strategy of a picture. This property makes the SA hypergraph automata better suited to model DNA-tile-based self-assembly systems.

SA-hypergraph automata provide a natural automata-theoretic model for patterned self-assemblies that will enable us to analyse self-assembly in an automata-theoretic framework. This framework lends itself easily to, e.g., descriptive and computational complexity analysis, and such studies may ultimately lead to classifications and hierarchies of patterned self-assembly systems based on the properties of their corresponding SA-hypergraph automata. An additional feature is that each SA-hypergraph automaton accepts an entire class of “supertiles” as opposed to a singleton set, which may also be of interest for some applications or analyses.

Acknowledgements

We thank Professor Grzegorz Rozenberg for extended discussions and his suggestion of applying hypergraph automata to the DNA self-assembly setting.

Bibliography

- [1] M. Anselmo, D. Giammarresi, and M. Madonia. Tiling automaton: A computational model for recognizable two-dimensional languages. In *CIAA*, pages 290–302. 2007.
- [2] R. D. Barish, R. Schulman, P. W. K. Rothmund, and E. Winfree. An information-bearing seed for nucleating algorithmic self-assembly. *Proceedings of the National Academy of Sciences*, 2009.
- [3] M. Blum and C. Hewitt. Automata on a 2-dimensional tape. In *SWAT (FOCS)*, pages 155–160, 1967.
- [4] A. Carayol and A. Meyer. Context-sensitive languages, rational graphs and determinism. *Logical Methods in Computer Science*, 2(2), 2006.
- [5] S. Crespi-Reghizzi and M. Pradella. Tile rewriting grammars and picture languages. *Theor. Comput. Sci.*, 340(1):257–272, 2005.
- [6] E. Czeizler and A. Popa. Synthesizing minimal tile sets for complex patterns in the framework of patterned DNA self-assembly. In *DNA Computing and Molecular Programming*,

volume 7433 of *Lecture Notes in Computer Science*, pages 58–72. Springer Berlin / Heidelberg, 2012.

- [7] L. de Prophetis and S. Varricchio. Recognizability of rectangular pictures by Wang systems. *Journal of Automata, Languages and Combinatorics*, 2(4):269, 1997.
- [8] D. Giammarresi and A. Restivo. *Two-dimensional languages*, pages 215–267. Springer-Verlag, 1997.
- [9] M. Göös and P. Orponen. Synthesizing minimal tile sets for patterned DNA self-assembly. In *DNA*, pages 71–82, 2010.
- [10] K. Inoue and A. Nakamura. Some properties of two-dimensional on-line tessellation acceptors. *Inf. Sci.*, 13(2):95–121, 1977.
- [11] D. Janssens and G. Rozenberg. Hypergraph systems generating graph languages. In *Graph-Grammars and Their Application to Computer Science*, pages 172–185, 1982.
- [12] A. Johnsen, M.-Y. Kao, and S. Seki. Computing minimum tile sets to self-assemble color patterns. In *ISAAC*, volume 8283 of *Lecture Notes in Computer Science*. Springer, 2013.
- [13] L. Kari, S. Kopecki, and S. Seki. 3-color bounded patterned self-assembly. In *DNA*, volume 8141 of *Lecture Notes in Computer Science*, pages 105–117. Springer, 2013.
- [14] T. Lempiäinen, E. Czeizler, and P. Orponen. Synthesizing small and reliable tile sets for patterned DNA self-assembly. In *DNA*, pages 145–159, 2011.
- [15] V. Lonati and M. Pradella. Picture recognizability with automata based on Wang tiles. In *SOFSEM*, pages 576–587, 2010.
- [16] V. Lonati and M. Pradella. Strategies to scan pictures with automata based on Wang tiles. *RAIRO - Theor. Inf. and Applic.*, 45(1):163–180, 2011.
- [17] X. Ma and F. Lombardi. Synthesis of tile sets for DNA self-assembly. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(5):963–967, 2008.

- [18] C. Morvan and C. Stirling. Rational graphs trace context-sensitive languages. In *MFCS*, pages 548–559, 2001.
- [19] P. W. K. Rothmund, N. Papadakis, and E. Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, page 2004.
- [20] E. Winfree. *Algorithmic self-assembly of DNA*. PhD thesis, 1998.

Chapter 4

Simplifying the Role of Signals in Tile Self-assembly¹

4.1 Introduction

A self-assembly system is an autonomous system whereby small components attach to each other via local interactions, in order to build a larger structure. Many examples of self-assembly systems exist in nature, a ubiquitous example being molecules that bind to each other via chemical bonds to form macromolecules or crystals. DNA-based self-assembly was introduced by Seeman [16] in 1982, when he designed a nanoscale DNA complex that could attach to four similar DNA complexes via DNA Watson-Crick complementarity. The self-assembly of many such DNA complexes resulted into a two-dimensional DNA lattice. Afterwards [18] [19] introduced rectangular DNA complexes called *tiles*, with a “sticky end” (single-stranded DNA sequences) at each corner, called “glue”. In this framework, a tile could attach to another tile if the sticky ends representing their corresponding corners were complementary DNA sequences. Winfree [17] introduced the abstract Tile Assembly Model (aTAM) as a theoretical model to describe DNA self-assembly systems. In the aTAM, labelled unit squares with coloured edges are used to represent DNA tiles. The colours (labels) on the edges represent the glues of the DNA tiles. In addition to the labels on its edges, each tile itself can have a label. In the aTAM,

¹This chapter is based on the paper Lila Kari, Amirhossein Simjour: Simplifying the Role of Signals in Tile Self-assembly. (Submitted)

each glue has a “strength” (a positive integer value), and a single tile can attach to an already assembled structure if the sum of the strengths of the glues which bind on the edges abutting the structure exceeds an *a priori* given positive numerical constant, called “temperature”.

Following Winfree’s aTAM, several other models were proposed for DNA self-assembly systems. For example, Aggarwal et al. [1] introduced a model that allows changes of the temperature during the assembly process, as well as a model that allows attachments of structures composed of multiple tiles. Aggarwal et al. [1] also compared different self-assembly models based on the number of different tile types (called *tile complexity* [15]) needed to construct an arbitrary rectangle. Reif et al. [14] investigated the possibility of tile detachment, and introduced a graph model as a general model for non-square tiles. Padilla et al. [10] introduced tiles that not only have multiple glues on each edge, but also have a limited ability to communicate with the tiles that are attached to them, via signals. The authors showed how signals can be implemented experimentally so as to be used to activate or deactivate glues. Doty et al. [6] expanded the aTAM model by adding the notion of repellent (negative-valued) glues. The authors showed that such a model can simulate a Turing machine using smaller intermediate assembly configurations compared to the original aTAM model. Other self-assembly models exist, see [12] for a review.

Padilla et al. [11] introduced the Signal Tile Assembly Model (STAM) as a mathematical model for tiles with the ability of sending signals that can activate and deactivate glues. The STAM uses as underlying model the 2-Handed Assembly Model (2HAM) [5], where two structures composed of multiple tiles can attach to each other if the strength of the glues at the abutting edges of the structures exceed the temperature (as opposed to the aTAM, where a single tile attaches to the growing structure at every time step).

Padilla et al. [11] proved that, for some specific shapes, the use of the STAM can reduce the tile complexity of the construction. Recently, Fochtman et al. [7] showed that a subset of the STAM that does not allow deactivation of glues can be simulated by three-dimensional 2HAM. Keenan et al. [9] investigated the usability of STAM in the replication of some patterns (rectangular structures with coloured tiles which form a pattern). The construction has some limitations, e.g., patterns are hole-free, and all tiles have to be able to support a set of predefined signals to make replication possible.

In this paper, we introduce DTAM (Detachable Tile Assembly Model), which is a simplified version of the STAM that is based on the 2-HAM model, and uses only glue-deactivating signals (instead of signals that can both activate and deactivate glues). We also introduce SDTAM (Simplified Detachable Tile Assembly Model), which is a further simplified version of DTAM wherein the attachment of only a single tile at each step is allowed. One of our main results shows a simulation of an arbitrary Turing machine by an SDTAM at temperature one (Theorem 4.3.1), showing thus that SDTAM can achieve universal computational power in spite of being a simplified version of STAM. Moreover, the Turing-simulating SDTAM we construct utilizes at most one signal per tile, and signals travel through only one tile before deactivating a glue, both of which could have implications for the practical implementations of such signal-based self-assembly systems. Our second result, Theorem 4.4.1, presents a DTAM construction of a “thin rectangle”, of size $N \times N!$, that uses only $O(\log N)$ tiles. This is an improvement over the tile complexity of existing models, the best of which use $O(\log N! / \log \log N!) = O(N)$ tiles to build the same rectangle [1].

The paper is organized as follows. Section 4.2 introduces the formal definitions of DTAM and SDTAM. In Section 4.3 we prove that SDTAM can simulate a Turing machine - the construction is based on simulating a deterministic zig-zag tile assembly system (known to be Turing universal [4]) by an SDTAM at temperature 1. Section 4.4 presents the construction of an $N \times N!$ rectangle using DTAM, and calculates its improved tile complexity, and Section 4.5 presents the conclusions.

4.2 The Detachable Tile Assembly Model

Informal Description of STAM

The Signal Tile Assembly Model (STAM)[11] is a tile assembly model based on 2HAM, wherein each tile possesses a set of glues on each edge (instead of one glue per edge, like in aTAM and 2HAM), and glues can be activated or deactivated by *signals*. In STAM, each glue on an edge can be in one of three states: *latent*, *on*, or *off*. Only a glue that is active (it is in the *on* state) can contribute to attaching the tile to another tile with an identical active glue

on the abutting edge. If the state of a glue is *off* or *latent*, the glue is inactive and it does not have any attachment power. In order to change the states of the glues, signals are used.

Intuitively, a signal is a mapping associated to a given tile that assigns to a glue on an edge a set of changes in the state of the glues on the other edges. For example, assume that tile t has glue g_e on the East edge, glue g_s on the South edge, and glue g_n on the North edge. Also assume that all these glues are *on*, and assume that there is a signal on the East side of the tile t that assigns a change of the state of the glue g_s to *off*. If that is the case, and if the tile t attaches to another tile via its East edge, the signal deactivates the glue g_s , that is, it changes its state to *off*. Signals can change the state of a glue from *latent* to *on* or to *off*, or from *on* to *off*. Note that, once a glue is in the *off* state, its state cannot be changed anymore. A tile can send a signal to its neighbour tile by activating a glue on an edge that is common with a neighbour tile. Signals can change the state of the glues, therefore signals can activate new glues and thus initiate a signal in the next tile. Moreover, signals can activate glues on a free edge and make new attachments possible. In addition to the activation, signals can deactivate the glues and, as a result, an existing structure might become unstable. In the STAM model, if the deactivation of a glue makes a structure unstable, the structure will break apart into two stable components.

Informal Introduction to DTAM

Here we define the *Detachable Tile Assembly Model (DTAM)* model as a variation of STAM. DTAM is a weaker version of STAM, whereby the signals are used only to *deactivate* glues. Thus all glues will start in the state *on*, and the state *latent* is not used. In addition, the signals themselves can only be turned *on*. Note that, since DTAM does not use the signals to activate glues, no new attachment possibilities (besides those that were present in the initial set-up) will be introduced during the self-assembly process. In Section 4.4 we will show that, in spite of these restrictions, the use of DTAM reduces the tile complexity of the construction of a thin rectangle as compared to [1].

We also introduce the *Simplified Detachable Tile Assembly Model (SDTAM)*, which is a restricted version of SDTAM wherein one starts with a single seed tile and, at each step, only the attachment of a single tile to the current configuration is allowed. In Section 4.3 we will

prove that SDTAM is Turing universal at temperature 1.

4.2.1 Formal Definition of DTAM

Detachable Tiles

A *detachable tile* is a unit square with the following properties. On each of its edges it has a set of *glues* and a set of *signals*, each of whom can be in a state from $Q = \{on, off\}$. Figure 4.1 part (i) shows an example of a tile in aTAM, and part (ii) illustrates the glues and signals of a detachable tile in DTAM. To each tile, we also associate a transition function, as described in the following. Note that, for simplicity, the states of the glues and signals are showed by superscripts: The superscript '+' indicates the state *on* and the superscript '-' indicates the state *off*. Since we only consider self-assembly models with detachable tiles, in the remainder of the paper we will call a *detachable tile* simply a *tile*.

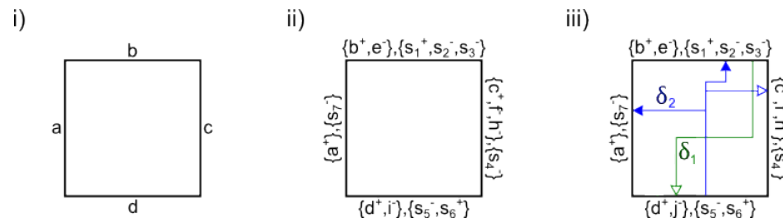


Figure 4.1: Part (i) shows a tile in the aTAM model: the tile has one glue on each side. Part (ii) shows the glues and signals on the edges of a detachable tile in the DTAM model: On the North edge, the presence of the set $\{b^+, e^-\}$ denotes that glue b is *on* and glue e is *off*, while the set $\{s_1^+, s_2^-, s_3^-\}$ denotes that the signal s_1 is *on*, and that signals s_2 and s_3 are *off* at this time. Part (iii) shows the tile defined in Example 0.1. The transition function Δ_t has two transitions. First, the transition $\delta_1 = \Delta_t(N, s_3) = \{(S, j)\}$ (green) starts from the signal s_3 on the North edge, and deactivates the glue j on the South edge. Second, the transition $\delta_2 = \Delta_t(S, s_5) = \{(E, c), (N, s_2), (W, s_7)\}$ (blue) starts from the signal s_5 on South edge, deactivates the glue c on the East edge, turns on the signal s_2 on the North edge, and turns on the signal s_7 on the West edge. The arrows with filled arrowheads represent the paths that turn signals *on*, while the arrows with empty arrowheads represent the paths that signals walk through to deactivate glues. All transitions are considered pending, and will be applied only if the originating signal (s_5 , respectively s_3) enters the *on* state.

Let Γ (the set of glues) and Σ (the set of signals) be two finite alphabets. The set of directions $D = \{N, E, W, S\}$ is the set of directions North, East, West, and South, respectively. If $d \in D$ is a direction, we define \bar{d} to be the opposite direction of d , where $\bar{W} = E$, $\bar{N} = S$, $\bar{E} = W$, and

$$\bar{S} = N.$$

A *tile* T over the alphabet $\Gamma \times \Sigma$ is a 3-tuple $t = (G_t, S_t, \Delta_t)$ where $G_t : D \rightarrow \mathcal{P}(\Gamma \times Q)$ is a function which, for every direction $d \in D$, specifies the set of glues on the edge d of the tile t , together with their respective states. Similarly, $S_t : D \rightarrow \mathcal{P}(\Sigma \times Q)$ is a function which, for every direction $d \in D$, specifies the set of all signals on the edge d of the tile t , together with their respective states.

Note that if, for a direction $d \in D$, we have that $G_t(d) = \emptyset$ (respectively $S_t(d) = \emptyset$), this means that there are no glues (respectively no signals) on the edge d of the tile t .

The *transition function* of the tile t is defined as a function $\Delta_t : D \times \Sigma \rightarrow \mathcal{P}((D \times \Gamma) \cup (D \times \Sigma))$. In DTAM, glues can only be deactivated, and signals can only be turned *on*, that is, $(d, g) \in \Delta_t(d', s')$ means that an active glue g^+ on the d edge of the tile will be deactivated (become g^-), and $(d, s) \in \Delta_t(d', s')$ means that an *off* signal s^- on the d edge of the tile will be turned *on* (become s^+).

Note that, for a tile t , the transition function Δ_t is invariant, but for each direction d , the states of glues $G_t(d)$, and the states of signals $S_t(d)$ change over time.

Transitions can only be applied when the corresponding signals are *on*. For example, the glue g^+ on the edge d of the tile t will be deactivated if and only if $(d, g) \in \Delta_t(d', s')$ and $(s', on) \in S_t(d')$. Similarly the signal s^- will be turned on if and only if $(d, s) \in \Delta_t(d', s')$ and $(s', on) \in S_t(d')$.

In addition, the tile t must satisfy the following conditions:

1. There can be only one glue and one signal of each kind on any given edge of a tile. In other words, for a given tile t , for all $d \in D$ and for all $(g, q), (g, q') \in G_t(d)$, we have $q = q'$. Similarly for signals, for a given tile t , for all $d \in D$ and for all $(s, q), (s, q') \in S_t(d)$, we have $q = q'$.
2. Transitions in Δ_t can only deactivate glues that already exist on an edge. This means that, if $(d, g) \in \Delta_t(d', g')$ then $(g, q) \in G_t(d)$ for some $q \in Q$.
3. Transitions in Δ_t can only activate signals that already exist on an edge. This means that, if $(d, s) \in \Delta_t(d', g')$ then $(s, q) \in S_t(d)$ for some $q \in Q$.

4. Transitions in Δ_t cannot deactivate their own starting signals. This means that $(d, s) \notin \Delta_t(d, s)$.

The following example, illustrated in Figure 4.1 part (iii), describes a detachable tile with the states of its glues, states of its signals, and its transitions. Note that, in this and subsequent figures, transitions are depicted by arrows: The origin (respectively the destination) of the arrow in the tile indicates the signal that must be turned on for the transition to be applied (respectively which signal is turned on by this transition, or which glue is deactivated by this transition). The convention that we use is that arrows depicting transitions which turn signals *on* have filled arrow-heads, while those depicting transitions that deactivate glues have empty arrowheads.

Example 0.1 Assume that $\Gamma = \{a, b, c, e, f, h, i, j\}$ and $\Sigma = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$. The tile t is defined as $t = (G_t, S_t, \Delta_t)$ where $G_t(W) = \{a^+\}$, $G_t(N) = \{b^+, e^-\}$, $G_t(E) = \{c^+, f^-, h^-\}$, $G_t(S) = \{d^+, j^-\}$, and $S_t(W) = \{s_7^-\}$, $S_t(N) = \{s_1^+, s_2^-, s_3^-\}$, $S_t(E) = \{s_4^-\}$, $S_t(S) = \{s_5^-, s_6^+\}$. The transition function Δ_t is defined by $\Delta_t(N, s_3) = \{(S, j)\}$ and $\Delta_t(S, s_5) = \{(E, c), (N, s_2), (W, s_7)\}$. Informally, the transition $\delta_1 : \Delta_t(N, s_3) = \{(S, j)\}$ starts from the signal s_3 on the North edge and activates the glue j on the South edge. Similarly, $\delta_2 : \Delta_t(S, s_5) = \{(E, c), (N, s_2), (W, s_7)\}$, starts from the signal s_5 on the South edge and deactivates the glue c on the East edge, turns on the signal s_2 on the North edge, and turns on the signal s_7 on the West edge. The transitions are considered pending, and will only be applied if the corresponding signals enter into the on state.

4.2.2 Transitions

Before the formal definition of *transitions* and *DTAM*, we review the definitions of *assembly-graph*, *glue strength*, *configuration*, and *supertile*.

Configurations and Associated Binding Graphs

Let $f: A \rightarrow B$ be a function and $A' \subseteq A$. The *restriction* of f to A' is a function $f|_{A'}: A' \rightarrow B$ defined as $f|_{A'}(x) = f(x)$ for all $x \in A'$.

A *pseudo-grid graph* is a directed weighted graph $G = (N_G, E_{G_v} \cup E_{G_h}, \pi_G)$ where N_G is a finite set of nodes, $E_{G_v}, E_{G_h} \subseteq N_G \times N_G$ are two sets of edges (called the vertical and horizontal edges respectively) such that $E_{G_v} \cap E_{G_h} = \emptyset$, and the function $\pi_G : (E_{G_v} \cup E_{G_h}) \rightarrow \mathbb{Z}^+$ is a weight function that associates a weight to every edge in the graph. The *node-induced subgraph* of G by the subset $N'_G \subseteq N_G$ is defined as the graph $G|_{N'_G} = (N'_G, E'_{G_v} \cup E'_{G_h}, \pi'_G)$ where $E'_{G_v} = \{(\alpha, \beta) \in E_{G_v} \mid \alpha, \beta \in N'_G\}$ and $E'_{G_h} = \{(\alpha, \beta) \in E_{G_h} \mid \alpha, \beta \in N'_G\}$, while π'_G equals $\pi_G|_{E'_{G_v} \cup E'_{G_h}}$.

If Γ is a set of glues, a *glue-strength* function g over Γ is defined as $g : \Gamma \rightarrow \mathbb{Z}^+$ where $\mathbb{Z}^+ = \{x \in \mathbb{Z} \mid x \geq 0\}$, and for a glue $\gamma \in \Gamma$, $g(\gamma)$ is called the *glue-strength* of the glue γ or shortly the *strength* of γ .

Let Θ be a set of tiles over $\Gamma \times \Sigma$, and let g be a glue-strength function over Γ . A (partial) mapping $C : \mathbb{Z}^2 \rightarrow \Theta$ is called a *general configuration* over the tile set Θ . Note that, in the experimental implementation of self-assembly by square DNA tiles that can bind to each other, tiles and supertiles can move on the two-dimensional plane during the process of assembly. To reflect this, if two general configurations coincide in all definition details, except that the domain of one can be obtained from the domain of the other via a translation, they will be considered to be the same.

A pseudo-grid graph $G_C = (N_G, E_{G_v} \cup E_{G_h}, \pi_G)$ is called the *assembly-graph associated to the general configuration C* over the tile set Θ , if there is a bijection $f : N_G \rightarrow \text{dom}(C)$ such that for all $\alpha, \beta \in N_G$, we have $(\alpha, \beta) \in E_{G_v}$ if and only if $f(\alpha) + (0, 1) = f(\beta)$, and $(\alpha, \beta) \in E_{G_h}$ if and only if $f(\alpha) + (1, 0) = f(\beta)$.

The weight function π_G of G_C is defined as follows. Assume, without loss of generality, that $e = (\alpha, \beta)$ is a vertical edge in E_{G_v} , from node α to node β , where $f(\alpha) = (x_1, y_1)$, and $C(x_1, y_1) = (G_1, S_1, \Delta_1) \in \Theta$ while $f(\beta) = (x_2, y_2)$, and $C(x_2, y_2) = (G_2, S_2, \Delta_2) \in \Theta$. The weight of an edge e of the assembly-graph G_C is defined as $\pi(e) = \sum g(\Gamma)$ where Γ ranges over all glues that are active on both the North edge of tile $C(x_1, y_1)$ and the South edge of tile $C(x_2, y_2)$. If there is no glue Γ with this property then $\pi(e) = 0$. The weight function is defined similarly for horizontal edges.

In other words, the nodes of an assembly-graph G_C associated to a general configuration C can be embedded in \mathbb{Z}^2 such that every edge in E_{G_v} has length 1 and points upwards, every

edge in E_{Gh} has length 1 and points rightwards, two nodes with Euclidean distance 1 are connected by an edge, and a tile is placed at each node. Note that graph edges are different from (perpendicular to) the edges of the tiles, with each graph edge corresponding to an attachment between two adjacent tiles, see Figure 4.2. The weight of a graph edge indicates the power of the attachment of corresponding adjacent tiles.

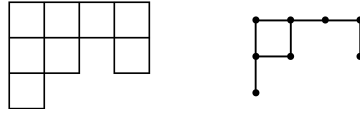


Figure 4.2: An example of a general configuration and the assembly-graph associated to it. All the edges of the graph are considered to be directed, with all the vertical edges pointing up, and all the horizontal edges pointing to the right.

The general configuration C over a tile set Θ is called a *connected configuration* or simply a *configuration* or *supertile* if the assembly-graph associated to C is connected. The configuration C is called *stable* at temperature τ (τ -stable supertile), if either

- For all i and $j \in \mathbb{Z}$, $C(i, j)$ is undefined except for one position (x, y) , where $C(x, y) \in \Theta$ or,
- The sum of the weights of the edges of each of the possible cuts of the assembly graph (all the edges between two partitions in the partitioning of the vertices of the assembly graph associated to C into two disjoint sets) associated to C is greater than or equal to τ .

A supertile C over Θ is called a *signal-stable supertile* if none of the transitions Δ_t of tiles t in C are applicable, moreover, the tiles in the configuration do not have any pending actions.

The self-assembly process proceeds by repeated applications of one of the following three types of transitions applied to either one supertile (detachment transition or action transition) or two supertiles (attachment transition) in the current set of configurations. If more than one transition is possible, then the order in which the transitions are applied is non-deterministic.

Attachment Transitions

During an attachment transition, two stable shape-compatible supertiles, with sufficient glue-strength at their abutting perimeter, assemble to form a larger supertile as shown in Figure 4.3

part (i) and (ii).

Formally, we say that the supertile V with associated assembly-graph $G_V = (N_G, E_{G_V} \cup E_{G_h}, \pi_G)$ is the result of an attachment of the two τ -stable supertiles V_1 and V_2 , if there exist two sets $P_1 \subset N_G$, $P_2 \subset N_G$, with $P_1 \cup P_2 = N_G$, and $P_1 \cap P_2 = \emptyset$, satisfying the following conditions:

1. $\text{dom}(V) = \text{dom}(V_1) \cup \text{dom}(V_2)$, $\text{dom}(V_1) \cap \text{dom}(V_2) = \emptyset$
2. $V(x, y) = V_1(x, y)$ for all $(x, y) \in \text{dom}(V_1)$, and $V(x, y) = V_2(x, y)$ for all $(x, y) \in \text{dom}(V_2)$, with the exception of any adjacent tiles at positions $(x_1, y_1), (x_2, y_2) \in \mathbb{Z}^2$ where $V_1(x_1, y_1) = t_1 = (G_1, S_1, \Delta_1)$, $V_2(x_2, y_2) = t_2 = (G_2, S_2, \Delta_2)$, $V(x_1, y_1) = t'_1 = (G'_1, S'_1, \Delta'_1)$, $V(x_2, y_2) = t'_2 = (G'_2, S'_2, \Delta'_2)$, where the following conditions hold:
 - The signal s^+ exists on the edge d of the tile t_1 that abuts t_2 , that is, $(s, \text{on}) \in S_1(d)$,
 - The signal s^- exists on the corresponding edge \bar{d} of the tile t_2 , $(s, \text{off}) \in S_2(\bar{d})$.

If these conditions are satisfied, then, the state of the signal s^- on the edge \bar{d} of t_2 is changed by this attachment transition to on , that is, $S'_2(\bar{d}) = (S_2(\bar{d}) \setminus \{(s, \text{off})\}) \cup \{(s, \text{on})\}$. All other elements of t_1 and t_2 remain unchanged.

3. The supertile V is stable at temperature τ .

Informally, condition (2) stipulates that if, during an attachment transition, two tiles attach and one of them has the signal s^+ on the abutting edge, then the signal s on the corresponding edge of the second tile also becomes on .

Detachment Transitions

During a detachment transition, a supertile in which the glue-strength along an internal “cut” is not strong enough (due to the deactivation of one or more glues), breaks into two smaller stable supertiles, as shown in Figure 4.3 part (v).

Formally, supertiles V_1 and V_2 are the result of a detachment transition applied to a supertile V with associated assembly graph $G_V = (N_G, E_{G_V} \cup E_{G_h}, \pi_G)$, if there exist two sets $N_1, N_2 \subset N_G$, such that $N_1 \cup N_2 = N_G$, $N_1 \cap N_2 = \emptyset$, and the following conditions hold:

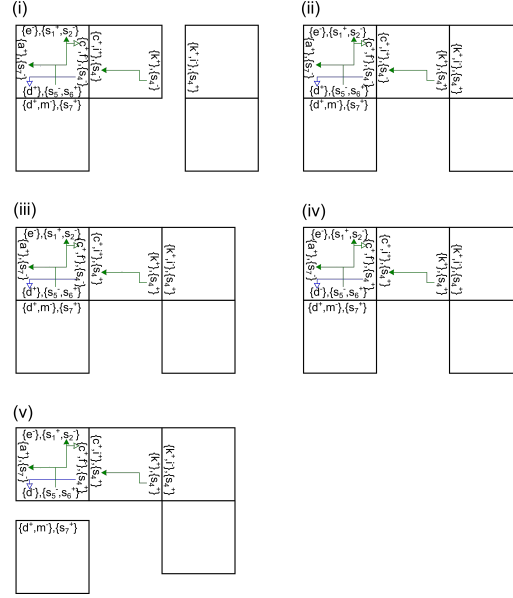


Figure 4.3: Parts (i) and (ii) show an example of the attachment transition: Two supertiles attach via the active glue k^+ (one assumes that the strength of the glue k is larger than or equal the temperature). Moreover, since the signal s_4^+ on the top-right tile of the newly formed supertile is *on*, the signal s_4^- on the abutting edge of its western neighbour tile is turned *on*, becoming s_4^+ . Parts (iii) and (iv) illustrate two consecutive action transitions: part (iii) - turning a signal *on*, part (iv) - deactivating a glue. First, since the top-middle tile contains the transition $(s_4, W) \in \Delta_1(E, s_4)$ (illustrated by the green arrow with filled arrowhead), and since the signal s_4 on the East edge of the top-middle tile is *on*, during the first action transition the signal s_4 on the West edge of the top-middle tile is turned *on*, becoming s_4^+ . Second, since the top-left tile contains the transition $(d, S) \in \Delta_2(E, s_4)$ (illustrated by the blue arrow with empty arrowhead), and since the signal s_4 on the East edge of the top-left tile is *on*, during this second action transition the glue d on the South edge of the tile is deactivated, becoming d^- . Part (v) illustrates a detachment transition: because there are no more active glues to hold together the top-left tile and the bottom-left tile of the supertile, two supertiles detach from each other.

1. $\text{dom}(V) = \text{dom}(V_1) \cup \text{dom}(V_2)$, $\text{dom}(V_1) \cap \text{dom}(V_2) = \emptyset$,
2. $V(x, y) = V_1(x, y)$ for all $(x, y) \in \text{dom}(V_1)$, and $V(x, y) = V_2(x, y)$ for all $(x, y) \in \text{dom}(V_2)$,
3. The weight of the cut (N_1, N_2) is smaller than τ , and $G|_{N_1}$ is the assembly-graph associated to V_1 , while $G|_{N_2}$ is the assembly-graph associated to V_2 .

Note that the supertiles V_1 and V_2 that result from a detachment transition are not necessarily τ -stable, and may further break into smaller supertiles if there exist cuts along which the attachments are not strong enough.

Action Transitions

The result of an action transition applied to the τ -stable supertile U is the supertile V , iff V coincides with U in all positions but one, where the tile in the supertile U appears *before* the application of the action transition, while the tile in the supertile V appears *after* of the application of that action transition. Note that there are two types of actions transitions: those that turn signals *on* (Figure 4.3 (iii)), and those that deactivate glues (Figure 4.3, (iv)).

Formally, a supertile V is the result of applying one of the transitions of one of the tiles of the τ -stable supertile U , iff for all $(x, y) \in \mathbb{Z}^2$, $V(x, y) = U(x, y)$ except two adjacent positions $(x_1, y_1), (x_2, y_2) \in \mathbb{Z}^2$ where $U(x_1, y_1) = t_1 = (G_1, S_1, \Delta_1)$, $U(x_2, y_2) = t_2 = (G_2, S_2, \Delta_2)$, $V(x_1, y_1) = t'_1 = (G'_1, S'_1, \Delta'_1)$, and $V(x_2, y_2) = t'_2 = (G'_2, S'_2, \Delta'_2)$, the tiles t_1 and t_2 are adjacent on the direction d of t_1 , and one of the following conditions holds:

- (glue deactivation) There exists a glue $g \in \Gamma$ and a signal $s \in \Sigma$ such that $(d', g) \in \Delta_1(d, s)$ and $(s, on) \in S_1(d)$. Then, $S_1 = S'_1$, $\Delta_1 = \Delta'_1$, $t_2 = t'_2$. The sets G_1 and G'_1 are the same except that $G'_1(d') = (G_1(d') \setminus \{(g, q) | q \in Q\}) \cup \{(g, off)\}$.
- (turning a signal *on*) There exist a signal $s \in \Sigma$ and a signal $s' \in \Sigma$ such that $(d', s') \in \Delta_1(d, s)$ and $(s, on) \in S_1(d)$. Then, $G_1 = G'_1$, $S_1 = S'_1$, $\Delta_1 = \Delta'_1$, $G_2 = G'_2$, $\Delta_2 = \Delta'_2$. The sets S_2 and S'_2 are the same except that, if $(s', q) \in S_2(\bar{d}')$, for some $q \in Q$, then $S'_2(\bar{d}') = (S_2(\bar{d}') \setminus \{(s', q) | q \in Q\}) \cup \{(s', on)\}$.

A *Detachable Tile Assembly System* based on *Detachable Tile Assembly Model (DTAM)* over alphabets Γ and Σ is a 5-tuple $(\Theta, g, \tau, \lambda, f)$, where Θ is a set of tiles over $\Gamma \times \Sigma$, g is a strength function over Γ , τ is a positive integer (temperature), λ is a finite set of starting τ -stable configurations, and $t_f \in \Theta$ is a single tile which will “mark” the final configuration of the assembly. The self-assembly process begins with the set of configurations $\lambda \cup \{\text{all configurations that consist of one tile}\}$ and proceeds by successive applications of the three types of transitions to either one configuration (in the case of a detachment or an action transition) or two configurations (in the case of an attachment transition) in the current set of configurations, asynchronously and non-deterministically. After applying each transition, the resulting configurations are be added to the current set of configurations. We say that a set of configura-

tions C is *derived from* $(\Theta, g, \tau, \lambda, f)$ if, starting from $\lambda \cup \{\text{all configurations that consist of one tile}\}$, we can obtain C by iteratively applying any of the three types of transitions. We call a supertile Z a *final supertile* if there exists a set of configurations C derived from $(\Theta, g, \tau, \lambda, f)$ with the property that $Z \in C$ and Z is a τ -stable signal-stable supertile such that there exist $i, j \in \mathbb{Z}^2$ with $Z(i, j) = t_f$.

Given positive integers M and N , we say that a detachable tile assembly system $(\Theta, g, \tau, \lambda, f)$ *assembles an $M \times N$ rectangle* if there exists a final supertile Z derived from this detachable tile assembly system, and there exist integers k_1 and k_2 , such that $Z(i + k_1, j + k_2) \in \Theta$ for all $1 \leq i \leq M$ and $1 \leq j \leq N$, while $Z(i, j)$ is undefined otherwise. Given positive integers M and N , we say that a detachable tile assembly system $(\Theta, g, \tau, \lambda, f)$ *uniquely assembles an $M \times N$ rectangle*, if it assembles an $M \times N$ rectangle and no other final supertile.

In comparing DTAM (the model we introduced in this section) with STAM, note first that, while DTAM has sets of signals separate from the sets of glues, these signals could easily be simulated in the STAM model as glues with strength zero. Thus, the introduction of the sets of signals as notation does not enhance the STAM model, that is, DTAM is not a generalization of STAM in this respect. The second observation is that in DTAM glues can only be deactivated and signals can only be activated: In this sense DTAM is a weaker version of STAM. Thirdly, since glues can only be deactivated, and signals can only be activated, the sets of pending actions are not needed in DTAM. Thus, while the formalism is slightly different, the DTAM model is a simplified version of STAM.

A *Simplified DTAM (SDTAM)* is a simplified version of DTAM, where all definitions are the same except those of the seed configuration and attachment transitions: In SDTAM, a seed configuration consists of a single seed tile, and the attachment of two supertiles is restricted to the case where one of the supertiles consists of a single tile.

4.3 Turing Universality

In this section we show that the SDTAM model can simulate a Turing machine at temperature $\tau = 1$. The basis of the proof is the simulation by SDTAM of any given *deterministic zig-zag tile assembly system* at temperature $\tau = 2$ – a type of self-assembly system that was proved in

[4] to be Turing universal. Moreover, our construction utilizes an SDTAM with at most one signal per tile, and where signals travel through only one tile before deactivating a glue.

A *deterministic zig-zag tile assembly system* at temperature $\tau = 2$ is a system that deterministically assembles a structure row by row, growing only upwards: Each row has to be completed before the next row starts, and each row grows in the opposite direction compared to the previous row. Moreover, the width of each row can only be greater than or equal to the width of the preceding row. For example, the first row could start from a seed tile and only grow to the right until a tile is placed that has a North glue of strength 2 and no East glue; then the second row starts assembling above this tile and has to grow to the left. See Figure 4.4 (i), where the order of tile placement is illustrated in that the directed path shows the direction of the growth of the assembly. Figure 4.4 (ii) illustrates the way in which attachment happens in such a system: Here, for any pair of tiles (t, t') , a small arrow enters tile t' from tile t if tile t was attached to the structure before tile t' , and tile t' attaches to the structure via the edge indicated by the arrow. Thus, incoming arrows in a tile indicate the edges by which the tile attached to the structure, and outgoing arrows indicate the edges by which the subsequent tile will attach. For example, the arrows of the top-right tile of Figure 4.4 (ii) indicate that the tile attached to the structure through its East and South glues, each of temperature 1, and that the next tile of the assembly will attach to the structure through the North glue of this tile, which has to have temperature 2 since only one edge is used for attachment. Figure 4.4 (iii) shows the 7 different ways in which tiles can attach in a temperature 2 deterministic zig-zag tile assembly system. We refer to [4] for a more detailed description.

In our construction, for a given deterministic temperature 2 zig-zag tile assembly system $\Phi = \langle T, s, 2 \rangle$, where T is a set of aTAM tiles, s is the seed tile, and 2 denotes the temperature, we define an SDTAM self-assembly system at temperature 1, namely $\Psi = (\Theta, g, \tau, \lambda, f)$ that simulates Φ . In this construction, each tile $t \in T$ will be replaced with a “gadget” of new tiles in Ψ that encodes the glues of t in the glues and shape of its borders. A “gadget” is a set of new tiles that simulate the behaviour of the original tile by uniquely assembling into a super-tile with the same attachment properties as the originating tile t .

Similar to [13] and [8] we will now simulate a deterministic zig-zag tile assembly model at temperature 2 with an SDTAM at temperature 1. Similar to the construction in [2] and [4], the

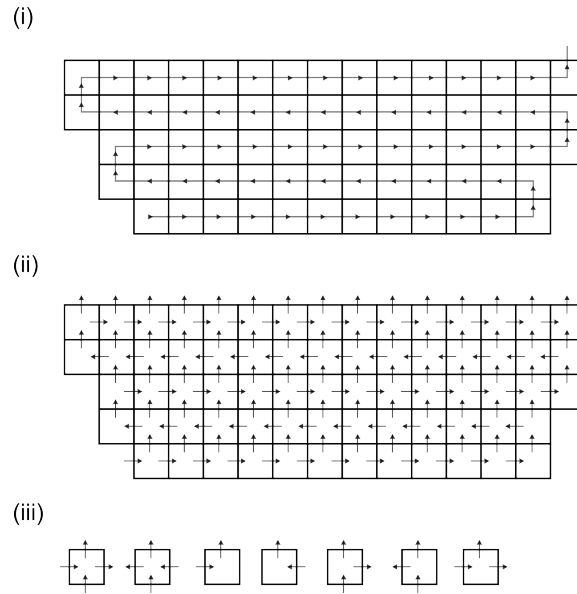


Figure 4.4: Simulation of a Turing machine using a deterministic zig-zag tile assembly system at temperature 2. The directed path in part (i) shows the direction of the growth of the assembly. Part (ii) illustrates the way in which attachment happens in such a system: Here, for any pair of tiles (t, t') , a small arrow enters tile t' from tile t if tile t was attached to the structure before tile t' and tile t' attaches to the structure via the edge indicated by the arrow. Part (iii) shows all the possible kinds of tile types in such a system, based on the way they attach to the structure.

simulation will be achieved by replacing each original tile by a “gadget” of new tiles. Since the new tile system operates at temperature 1, this construction alone is not sufficient to guarantee the growth of each gadget uniquely, and incorrect attachments can form. Behsaz et al. [2] employed staged tile assembly system to control the assembly growth. In our construction, we will use signals that lead to the detachment of incorrect growths to solve the same problem.

Theorem 4.3.1 *For any deterministic temperature 2 zig-zag tile assembly system $\Gamma = \langle T, s, 2 \rangle$ there exists a simplified detachable tile assembly system SDTAM at temperature $\tau = 1$ that simulates Γ with horizontal scale $O(\log(|T|))$.*

Proof The proof is by construction. Each tile in the tile set T is replaced by a set of new tiles with the property that they self-organize uniquely into a supertile which we will call a “gadget”. Figure 4.5 shows the original tiles and their corresponding gadgets made out of new tiles. The arrows indicate the path that describes the order in which the self-assembly of the gadget proceeds, with the observation that the two bottom “legs” of each gadget self-assemble independently of this path.

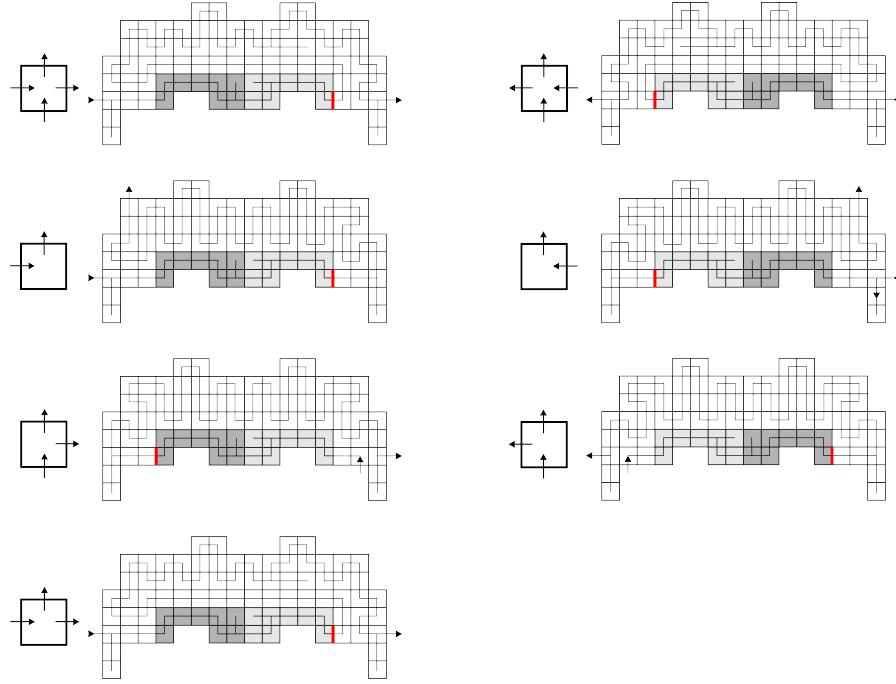


Figure 4.5: In each of the two columns, a tile on left is an original tile from the deterministic zig-zag tile assembly model at temperature 2, and its corresponding gadget made out of SD-TAM tiles is shown at its immediate right. For example, the top left tile with North glue 00 (right/right) and South glue 10 (left/right) is simulated by the gadget immediately next to it. This gadget has on its North side two bumps both positioned at the right of their respective two-level blocks. The gadget has at its South two dents, the first positioned at the left and the second positioned at the right of their corresponding two-level blocks. The arrows on the gadget show the order of growth of the assembly.

Consider for example the original tile and its corresponding gadget situated at the top-left of Figure 4.5.

The East and West glues of the original tile are simulated as follows. The West glue of the original tile becomes the West glue of the first (new) tile of the path in the corresponding gadget, indicated by the arrow that enters the gadget, see Figure 4.5, top-left. Similarly, the East glue of the original tile becomes the East glue of the (new) tile of the corresponding gadget indicated by the arrow that exits the gadget.

Since the new tile system is a temperature 1, the fact that sometimes two glue matching sides are needed for an attachment cannot be simulated directly, and a combination of bumps-and-dents will be needed to create the same effect. To that end, the North and South glues of the original tile are simulated as follows.

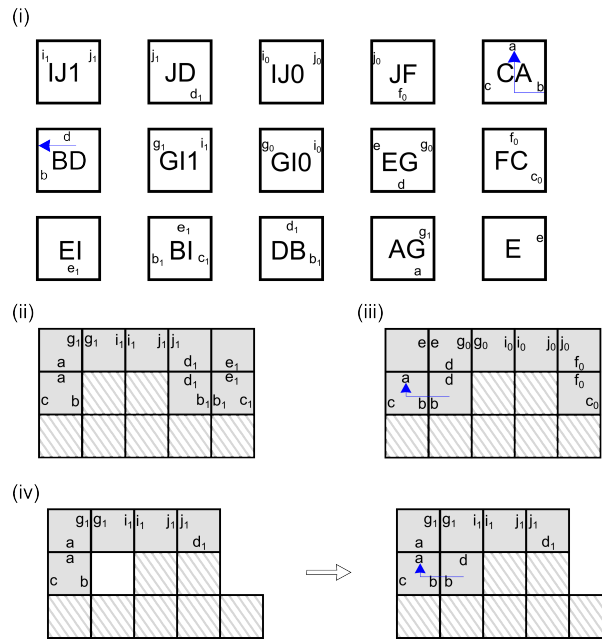


Figure 4.6: The (new) tiles in part (i) are used in the bit reader (South glue bit of a gadget) in the construction in Theorem 4.3.1. Part (ii) shows the construction of the bit reader when the bump of an incoming bottom gadget indicates bit 1. Part (iii) shows the construction of the bit reader when the bump of an incoming bottom gadget indicates bit 0. Part (iv) shows an example of an incorrect growth, and how it will be fixed. As shown in part (iv), incorrect attachments cannot grow beyond the bump and will stop. Afterward, tile BD can attach to the east edge of the tile CA to detach the incorrect attachments. For readability, in parts (ii), (iii), and (iv), the tile labels have been omitted.

The glues on the North and South edges of t are first encoded as binary numbers. Since there are no more than $4 * |T|$ glues, we only need to use numbers from zero to $4 * |T|$ for the coding, and only $\log(4 * |T|)$ bits are needed to encode these numbers. To simulate the North glue of the tile t , each bit in the binary number encoding that glue is encoded as a *two-level block with a bump*, consisting of seven new tiles (5 tiles in the bottom row and 2 tiles in the top row). If the two tiles in the top row are located on top of the second and third tile of the bottom row, that is, the bump is positioned at the left, then the bit is 1 (see Figure 4.6 (ii), the hashed tiles). Similarly, if the two tiles in the top row are placed on top of the third and fourth tile from the bottom row, that is, the bump is positioned at the right, then the bit is 0 (see Figure 4.6 (iii), the hashed tiles).

Similarly, the glue on the South edge of t is encoded as a binary number implemented by a sequence of two-level blocks with dents (that will geometrically fit into blocks with matching

bumps), as follows. Each bit in the binary number representing the South glue of t is encoded by a *two-level block with a dent*, consisting of eight tiles (5 tiles in the top row and 3 tiles in the bottom row). If the three tiles in the bottom row are located under the first, forth, and fifth tile in the top row, that is, the dent is positioned at the left, then the bit is 1 (see Figure 4.6 (ii), the grey tiles). Similarly, if the three tiles in the bottom row are placed under the first, second and fifth tiles from the top row, that is, if the dent is positioned at the right, then the bit is 0 (see Figure 4.6 (iii), the grey tiles).

Figure 4.5 illustrates some tiles and the corresponding gadgets made out of new tiles, that simulate them. For example, the top-left tile with North glue 00 and South glue 10 is simulated by the gadget at its immediate right.

Note that, in each gadget in Figure 4.5, the (dark and light) grey portions implement the portions that encode the South glue. These portions acts as “bit-readers” in the sense that they “read” the bits that encode the North glue of a gadget that could attach to it from the South. Figure 4.6 (i) shows the (new) tile types used to encode the bit readers (i.e. South glue). Note that if a tile edge has no label, this means there is no glue on that edge and no attachment can form. The tile can still attach to a structure, via another edge, and glue mismatches are allowed.

Figure 4.6 (ii) shows bit 1 on the North part of a gadget (hashed) and its corresponding bit reader, i.e. the South glue of a gadget that will fit on top of it (grey). Assuming that the bottom gadget with its hashed portion representing bit 1 is already assembled, the only tile that can attach to it is the one labelled CA, through glue b . After that, the tile labelled AG is the only possible attachment, followed by the attachments of tiles labelled GI1, IJ1, JD, DB, BI and EI, in this order. Similarly Figure 4.6(iii) shows bit 0 (hashed) of a gadget, and its bit reader, i.e., South glue of a gadget that will fit on top (grey). Assuming that the bottom gadget with its hashed portion representing bit 0 is already assembled, the self-assembly proceeds with BD, CA, EG, E, GI0, IJ0, JF, FC.

Note here that if the West glue of the original tile is labelled c , that glue is transmitted to the bit reader of the corresponding gadget. After all the bits of the South glue of the gadget are formed, the last tile of the bit reader contains in its East glue (marked in red in Figure 4.5, top-left gadget) information about both the West glue of the original tile, and about its South glue. This information is encoded in the label of the glue, which is c_1 if the West glue was c and

the South glue was 1, and c_0 if the West glue was c and the South glue was 0 (see Figure 4.6(i)).

Incorrect attachments during the assembly of the bit reader (South glue) may form, due to the fact that there may exist original tiles that have the same West glue but different South glues (bit readers). Figure 4.6(iv) shows how an incorrect attachment may begin to form, and how it is corrected by the use of signals which will detach the incorrect growth. For example, assume we are reading bit 0 (Figure 4.6(iii)), but the bit reader attempts to read it as a 1 (Figure 4.6(ii)). After the attachment of tile CA, both tile AG and BD can attach to the current configuration. If, instead of tile BD (which would proceed to correctly read bit 0), tile AG incorrectly attaches, then tiles GI1, IJ1, and JD will attach to the corresponding positions. After the attachment of the tile JD, since the bump representing bit 1 occupies the South side of JD, no further attachment to JD is possible, and the growth of the structure in this direction stops. In order to repair this incorrect growth, when tile BD attaches to the hole on East side of the tile CA, it activates the signal on the east edge of CA. This signal deactivates the glue a on the north edge of the tile CA and detaches the incorrect growth, which can now be replaced by the bit reader for bit 0.

Due to the fact that the zig-zag self-assembly systems that we are simulating are deterministic, once the West glue and the South glue of a tile are known, the other two glues are uniquely determined, so the rest of the white tiles in the gadget in Figure 4.5 top-left can be hard-coded (this means that the tile at each position is unique and has glues that make sure that the tile can only attach at that particular position).

One can similarly construct gadgets simulating all the other tile types of the given deterministic zig-zag tile system at temperature 2. From the construction it follows that the function that maps each original tile to a set of new tiles that self-organize into the corresponding gadget is a one-to-one correspondence. Since deterministic zig-zag tile systems at temperature 2 are Turing universal, it follows that SDTAM is also Turing universal at temperature 1.

Regarding the scale of the construction, assume that the original tile set has $|T|$ tile types and their glues which, as seen previously, each necessitates $\log(4 * |T|)$ bits to encode. Each of the original tiles is simulated by a gadget. The width of such a gadget consists of 3 extra tiles on each side, plus 5 tiles for each bit, for a total of $(3 + 5 * \log(4 * |T|) + 3)$ tiles. Thus, if the width of a structure assembled in the original deterministic zig-zag tile assembly system at

temperature 2 is W , then the width of the structure assembled by SDTAM tiles at temperature 1 that simulates it is $(3 + 5 * \log(4 * |T|) + 3) * W$, that is, $O(\log(|T|) * W)$.

If we ignore the right and left leg of each gadget, and the North bumps (which will interlock with the gadget above it and with the one underneath it, and thus do not add to the height), then the height of a simulating gadget is 5 tiles. If the height of the original zig-zag structure at temperature 2 was H , then the height of the new structure at temperature 1 that simulates it will be $(2 + 5 * H + 1) = O(H)$ (the constants account for the height of the legs of the gadgets on the first row of the new structure, and the height of the North bumps of the gadgets on its top-most row, respectively).

From the construction above it also follows that, besides being Turing universal, this SDTAM utilizes at most one signal per tile, and signals travel through only one tile before deactivating a glue.

4.4 Self-assembly of Thin Rectangles

In this section, we present a DTAM tile assembly system that, for a given $N > 6$, uniquely assembles an $N \times N!$ rectangle, and uses only $O(\log N)$ tile types. Throughout the proof we will assume that the DTAM self-assembly system that we construct is at temperature $\tau = 6$ and that the strength of each glue is either $\tau/2 = 3$ or $\tau/3 = 2$.

4.4.1 Informal description of the DTAM tile assembly system

We start with a high level explanation of our construction. Figure 4.7 shows the method that can be used to build an $(M + 1) \times ((M + 1)! + 2C)$ rectangle starting from an $M \times (M! + 2C)$ rectangle, for a given $M > 6$. This method can then be used for the construction of an $N \times N!$ rectangle as follows.

Start with an initial $6 \times (6! + 2C)$ rectangle, where $C = \lceil \log(N - 5) + 2 \rceil$, and use the method (illustrated in Figure 4.7) to add a row and an appropriate number of columns. C is the width of each of the two dark grey borders of the rectangle in Figure (4.7a), which control the

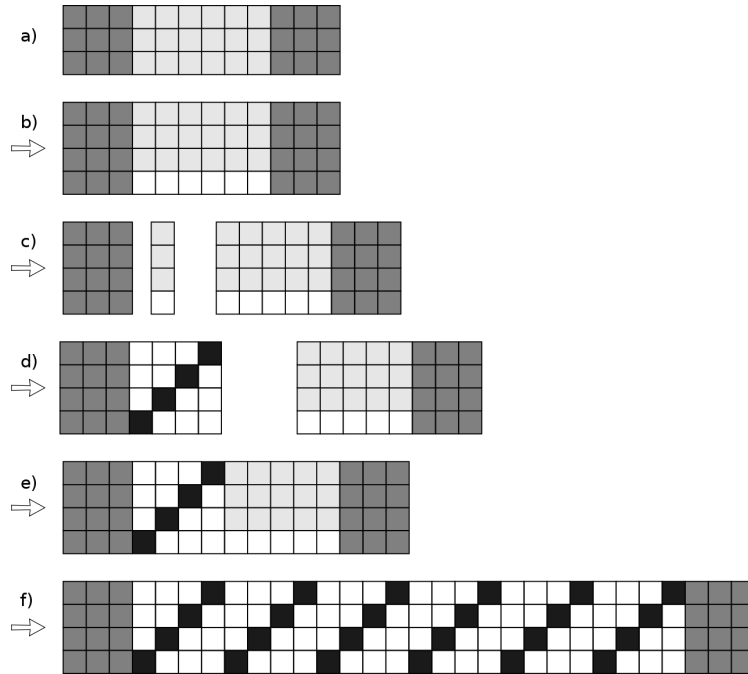


Figure 4.7: This figure illustrates the idea of the steps that replace an $M \times (M! + 2C)$ rectangle (7a) by an $(M + 1) \times ((M + 1)! + 2C)$ rectangle (7f), for $M = 3$ and $C = 3$. (7b) illustrates the addition of a new row to the rectangle in (7a). (7c) illustrates the detachment of the leftmost light grey column. (7d) illustrates the replacement of one light grey column by $(M + 1)$ columns of the same shape and size. (7e) illustrates the reattachment of the left part of (7d) to the right part. Repeating steps (7b) to (7e) for all the $M!$ light grey columns results into an $(M + 1) \times ((M + 1)! + 2C)$ rectangle that is shown in (7f).

assembly process but will not be retained in the final structure. Repeat the steps illustrated in Figure (4.7b)-(4.7e) for increasing values of M , until an $N \times (N! + 2C)$ rectangle is obtained. At the end, remove the C columns from the left and C columns from the right to obtain the desired $N \times N!$ rectangle.

Figure 4.7 illustrates the method used to obtain an $(M + 1) \times ((M + 1)! + 2C)$ rectangle from an $M \times (M! + 2C)$ rectangle for the case $M = 3$ and $C = 3$. Note that this is an illustration of the general idea of the method, and not a real example, for which N (and M) should be greater than 6. The input is a rectangle of size $M \times (M! + 2C)$, see Figure (4.7a).

Step 1. One row is added to the bottom of this rectangle, as illustrated in Figure (4.7b).

Step 2. The leftmost white tile of the rectangle in Figure (4.7b) sends a signal to all the tiles above it, resulting in the detachment of this light grey column from the rest of the rectangle.

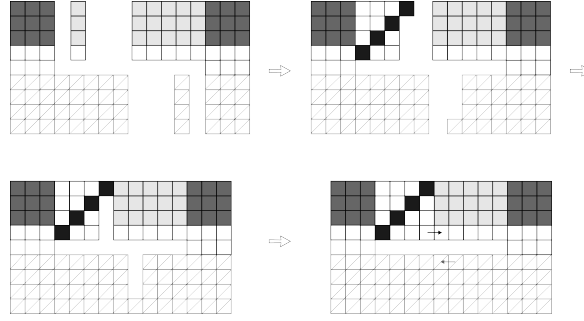


Figure 4.8: This figure illustrates the use of the mirror rectangle (hashed) in the process detailed in Figure (4.7c) through (4.7e). The mirror rectangle is used to make sure that only the corresponding right and left parts in Figure (4.7d) can attach to each other (not pieces of rectangles from different steps), and prevent any incorrect attachments by imposing additional geometrical constraints on the attachment.

As a result, the rectangle is now divided into three parts as seen in Figure (4.7c). (The middle column that was detached in this step is waste, and will not be used anymore.)

Step 3. The remaining left part of the rectangle (its left dark grey portion) expands by $(M + 1)$ columns, Figure (4.7d).

Step 4. The expanded part (the left part of Figure (4.7d)), and the right part of Figure (4.7c) reattach to each other, see Figure (4.7e).

Step 5. When the two parts reattach, the tile that was the bottom-right tile of the left part of Figure (4.7d) sends a signal eastwards, to start the detachment of the next light grey column.

Step 6. The above four steps are repeated for all the light grey columns in Figure (4.7c), resulting in Figure (4.7f).

The output is a new rectangle of size $(M + 1) \times ((M + 1)! + 2C)$. In the case of our example the output is the $4 \times (4! + 2 \cdot 3)$ rectangle in Figure (4.7f).

In order to prevent incorrect attachments in the process illustrated in Figure 4.7, we will need another rectangle, of the same dimensions (see Figure 4.8, the hashed rectangle at the bottom of each subfigure). This second rectangle is built using a functionally equivalent but disjoint tile set. This means that, if the tile set Θ was used to build the first rectangle as well as the row underneath it, then a mirror tile-set set Θ' can be used, whose tiles are obtained by

flipping each of the tiles from Θ about their top-left/bottom-right diagonal but using, instead of its original glues, the corresponding glues from a glue set that is isomorphic to the set of glues of the tiles in Θ . As a result, the tiles from Θ' will self-assemble into a copy of the first rectangle, flipped about its top-left/bottom-right diagonal. We will call this the “mirror rectangle”. The tiles from Θ' will not interact with tiles from Θ . (There are some exceptions, namely some glues that are common to Θ and Θ' , that will be explained later in this section.)

The mirror rectangle attaches to the one we wish to expand (Figure (4.7a)) at the end of the step in Figure (4.7b), and will stay connected to it for the duration of the expansion. The mirror rectangle expands alongside with the original one, as it has mirror tiles that can self-assemble in the same way.

The purpose of the mirror rectangle is to make sure that in the step from Figure (4.7d) to Figure (4.7e), only the left part and the right part of the rectangle in Figure (4.7d) reattach to each other, and not other partial rectangles.

When all the columns are expanded, the top rectangle and its mirror rectangle detach from each other.

After adding $(N - 6)$ rows, the dark grey parts (which encode a counter) stop the self-assembly. Finally, the dark grey columns detach, and only the desired $N \times N!$ rectangle remains.

4.4.2 Detailed description of the DTAM tile assembly system

Theorem 4.4.1 *For any given $N > 6$ there exists a DTAM tile assembly system at temperature $\tau = 6$, with glues of strengths either $\tau/2$ or $\tau/3$, that uniquely assembles an $N \times N!$ rectangle and uses only $O(\log(N))$ tile types.*

Proof: By construction. The starting configuration of the system is a $6 \times (6! + 2C)$ rectangle, where $C = \lceil \log(N - 5) + 2 \rceil$, and which satisfies the properties (detailed in the sequel) of the input rectangle for Step 1. We now describe in detail the steps of the method used to obtain an $(M + 1) \times ((M + 1)! + 2C)$ rectangle from an $M \times (M! + 2C)$ rectangle.

Step 1 (Figure 4.9 (i), (ii), (iii), (iv)): Addition of a new bottom row, and of the mirror rectangle.

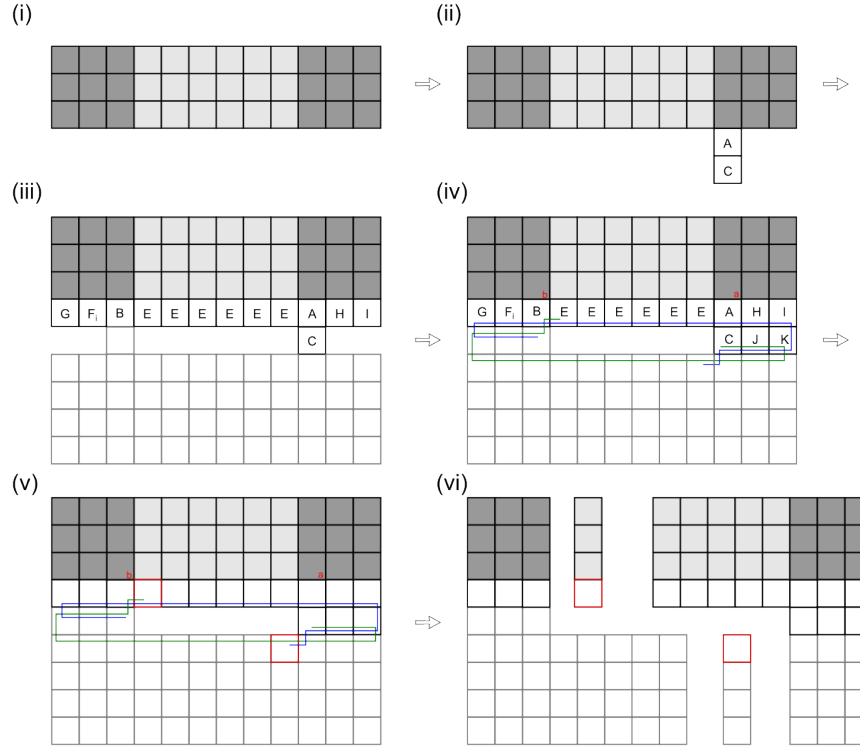


Figure 4.9: Illustration of Step 1 and Step 2. (i) is the input rectangle, (ii) shows the attachment of tiles A and C, (iii) shows the addition of the new bottom row and the mirror rectangle, (iv) shows the paths of the detachment signal that is initialized by tile C (green), and its mirror signal path (blue), (v) shows the tiles that are the final recipients of the detachment signals (outlined in red), and (vi) shows the detachment of one column from the input rectangle and of the mirror column from the mirror rectangle.

The rectangle we start from consists of C columns on the right side, and C columns on the left side, in addition to the $M!$ columns in the middle. In addition, the rectangle must satisfy the following properties:

- The glues on the south edge of the rectangle follow the pattern shown in Figure 4.10. (Detailed definitions of the tile types and their glues are shown in Figure 4.11.)
- The tiles in the white area are connected to their east and west neighbours only with the glues from the set $\{d, d_1, d_2, x, m_1, m_2\}$. (Detailed definitions of the tile types and their glues are shown in Figure 4.11.)

To this thin rectangle, a new row will be attached at the bottom, as described below.

The tiles from the tile set $\Theta_1 = \{A, B, C, E, F_0, F_1, F_2, F_3, F_4, G, G', H, I, J, K\}$, shown in the first three rows of Figure 4.11, attach a new row to the bottom of the rectangle, as well as

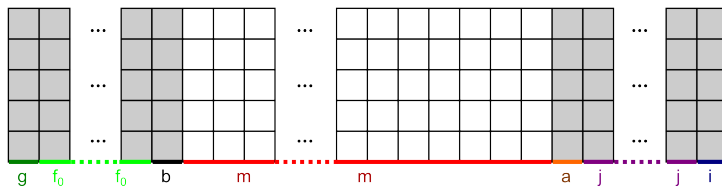


Figure 4.10: A thin rectangle that is the input of Step 1 of the construction, and the glues on the tiles of its bottom edge. (Detailed definitions of the tile types and their glues are in Figure 4.11)

an extra tile (C) underneath it, that connects it to the mirror rectangle. In addition, tiles J and K fill in the positions at the right of tile C (see Figure 4.9(iv)), as follows.

The tiles C and B are the ones that enable the mirror rectangle to attach under the original rectangle. Their glues b' and c' are mapped to the glues c' and b' , respectively, of mirror tiles in the mirror tile set. We define the power of these two glues to be $\tau/2$. As a consequence, the mirror rectangle can attach to the original one using only tiles B and C , through their two glues c' and b' , as $\tau/2 + \tau/2 = \tau$ (see Figure 4.9(iii)).

In order to prevent the detachment of the mirror rectangle from the original one in the next step (Step 2 - column detachment), this connection has to be stronger than τ . This is achieved by adding two tile types, J and K , with north glues of strength $\tau/3$. Tiles J and K attach to the structure after the mirror rectangle is completely attached. Their addition makes the connection between the right part of the original rectangle and the mirror rectangle have total strength $\tau/2 + (C - 2)\tau/3 + \tau/3 > \tau$ (through tile C , followed by $C - 2$ copies of tile J , followed by tile K), see Figure 4.9(iv). Due to the construction of the mirror tile set tile, mirror tiles of the tiles J and K exist, that will ensure that the connection between the left side of the original rectangle and the mirror rectangle becomes also $\tau/2 + (C - 2)\tau/3 + \tau/3 > \tau$.

Note that each of these set of bonds has a total strength greater than τ . Thus, when the entire structure breaks into two (a left part and a right part) during the next step, the top piece (from the original rectangle) and the bottom piece (from the mirror rectangle, directly underneath it) in each of these parts remain attached together.

Step 2 (Figure 4.9 (iv), (v), (vi)): Column detachment.

The column detachment is initiated by tile C . Signal s_1 from tile C (shown in green in

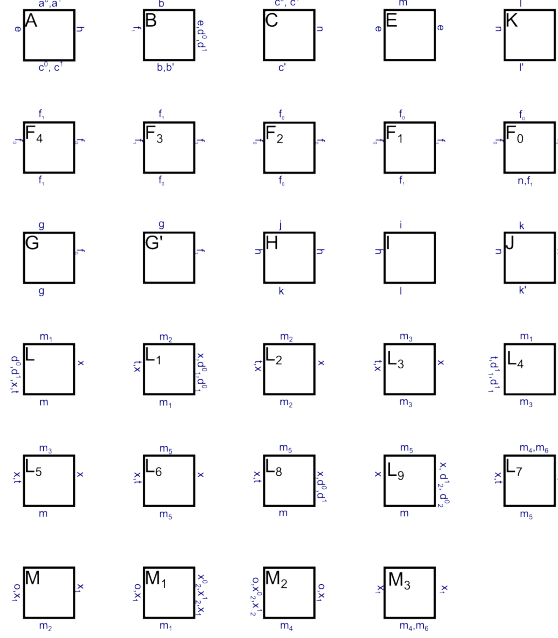


Figure 4.11: The tile types that are used in the construction of a thin rectangle and their respective glues (signals are not shown in this figure). The tile types in the top three rows are used in Step 1, and the tile types in the bottom three rows are used in Step 3. The power of all the glues is $\tau/2$, where τ is the temperature of the DTAM system, except for the glues t , k , k' , l and l' whose power is $\tau/3$. (Note that glue superscripts 0 and 1 are only used in this figure, to simplify the explanation for the glue strengths. That is, the presence of d^0 and d^1 on the same edge only indicates that glue d is present, and that its strength is the sum the individual strengths of its components d^0 and d^1 , both of which have strength $\tau/2$). Functionally, glues x , o , t , d , d_2 , d_1 are used for the attachment of columns, while glue m with different subscripts is used for the attachment of rows. Glue f with different subscripts is used for the attachment of the counter. Table 4.1 and Table 4.2 include the detailed definition of the signals and transitions for all these tile types.

Figure 4.9(iv)) travels through tiles J and K , continues through the top row of the mirror rectangle, and then travels through the tiles under G , F_1 and B (the mirror tiles of C , J and K), reaching tile E (marked in red in Figure 4.9(v)). Similarly, the mirror tile of C (the tile under B) sends the signal s_2 (shown in blue in Figure 4.9(iv)), which travels a symmetric path and reaches the mirror tile of E (marked in red in Figure 4.9(v)). If these two signals reach their destination, this guarantees that the two portions that connect the original rectangle with the mirror rectangle are filled in, with no holes. This further implies that the subsequent column detachment will not result, inadvertently, in the separation of any part of the original rectangle from its corresponding part in the mirror rectangle (the total strength of the attachments that

keep them together is larger than τ).

The column detachment is illustrated schematically in Figure 4.9(v) and (vi). When the detachment signal reaches tile E , this tile deactivates the glue on its west edge. Moreover, tile E sends a signal to its north edge, which travels northwards. As a result, all the tiles above tile E , in the same column, deactivate their attachment with their west tile. In order to deactivate the east side of the column, tile E sends another signal to the tile on its east side. This signal goes through all the tiles above it, and deactivates their west attachments. As a result of this process, the column becomes detached from the original rectangle and becomes waste. A similar process, which starts when the signal reaches the mirror tile of E , happens in the mirror rectangle. As a result, a mirror column becomes detached from the mirror rectangle, as seen in Figure 4.9 (vi).

The details of the process are illustrated in Figure 4.12. In order to prevent undesired attachments, instead of first deactivating all the west glues of the column tiles, and then deactivating of all its east glues, we alternate. That is, starting with the bottom column tile E , we first detach the west glue of E , then its east glue, as well as send a signal to the tile east of E , deactivating its west glue. As tile E detaches and leaves a hole, the hole is immediately filled by a new tile, L . This new tile is part of the first new column generated during the expansion process of Step 3, and this new tile sends a signal to the tile above it, so as to continue the column detachment. Combining column detachment with the attachment of the first column of the subsequent expansion step, together with the signals that deactivate the glues of the tiles situated east of the column being detached, guarantee that no incorrect attachments can occur. This is because, during the expansion process, when the right and the left parts of the rectangle are separated, the left part will not have any east glue that matches an active west glue from the right part and that could have potentially led to an incorrect attachment.

Step 3 & 4 (Figure 4.13 and 4.14): Expansion and re-attachment.

During this step, the left part of the rectangle (as well as the mirror rectangle) is expanded by $(M + 1)$ columns. Figure 4.11 (last three rows) shows all the tile types that are needed to expand a rectangle with new columns.

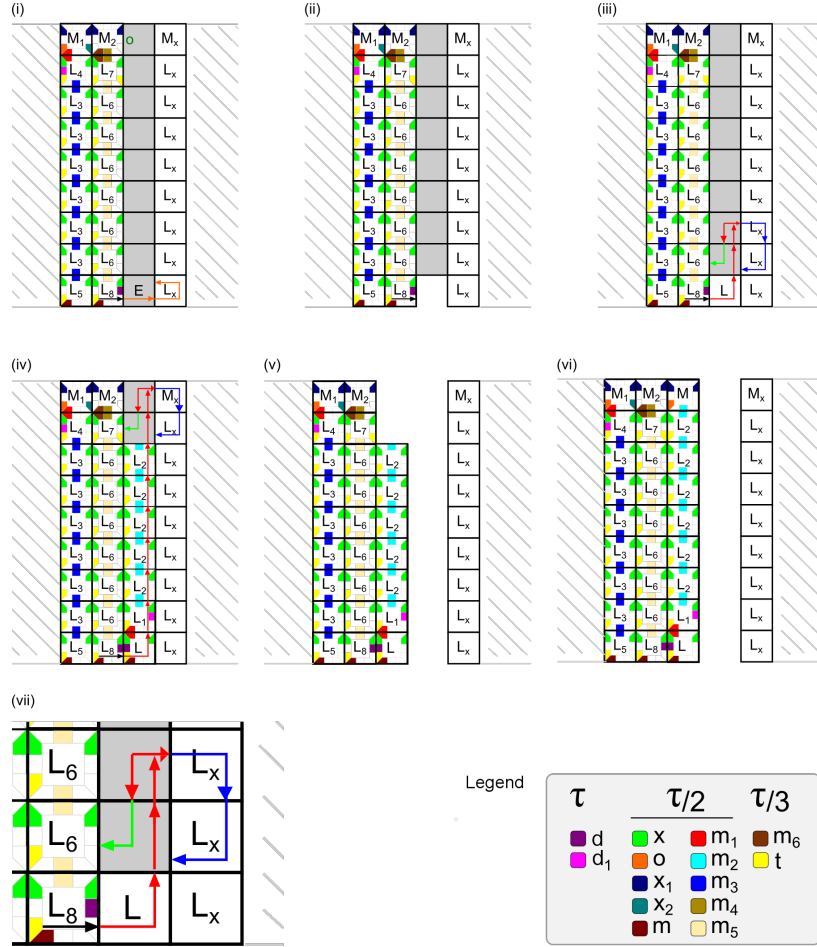


Figure 4.12: A detailed view of Step 2, and the attachment of the first column in Step 3, that were illustrated schematically in Figure 4.9(iv): (i) - tile L_8 initiates a signal to detach tile E at the bottom of the first light grey column (marked in red in Figure 4.9(iv)); (ii) - tile E is detached; (iii) - tile L attaches in the place of tile E and sends a signal to the tile above it, deactivating its west glue; (iv) - all the tiles above tile L are detached and are replaced by tiles from the first column of Step 3; (v) - the left and right part of the rectangle(s) detach from each other; (vi) the first column of Step 3 is completed; (vii) enlarged copy of the bottom rows of part (iii).

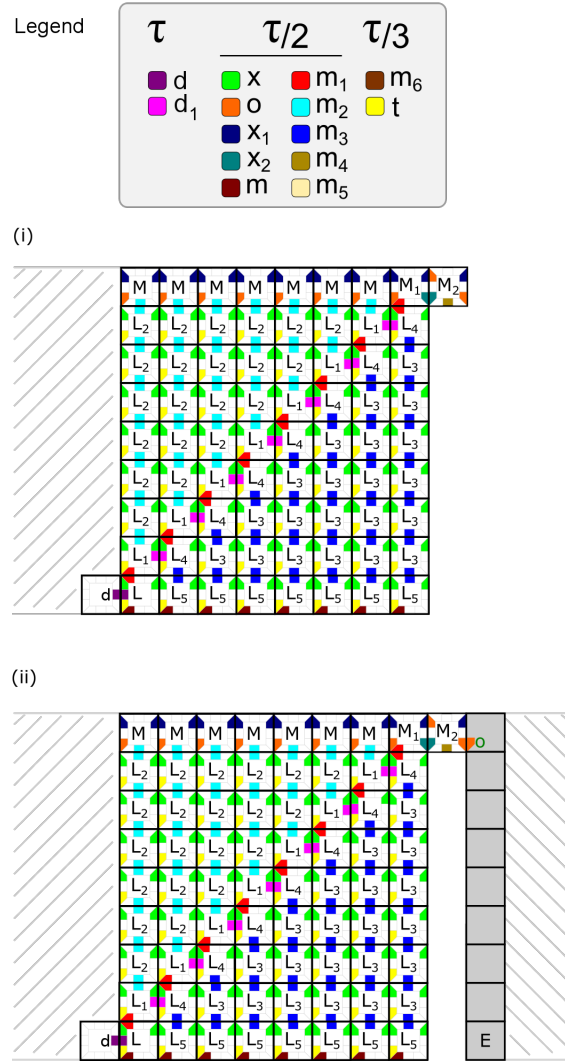


Figure 4.13: Together with the next figure (Figure 4.14), this figure illustrates Step 3 (expansion), and Step 4 (reattachment). The top of the figure lists the colour coding of the glues of tiles involved in these steps, and their strength - the actual glues were listed in Figure 4.11. (i) shows the construction of the square that starts from tile L (bottom left) and adds the columns of the square one by one, from left to right, ending in tile M_2 (top right); (ii) shows the reattachment of the right part of the rectangle to its left part, through tile M_2 and its mirror tile in the mirror rectangle (the latter is not shown here).

The construction started already in Step 2 by the attachment to the left part of the rectangle of tile L , via its west edge (this tile replaces tile E). Subsequently, a square will self-assemble that will attach to the east of the left part of the rectangle (see Figure 4.13 (ii)). Since the rectangle (that now includes the added bottom row) has height $(M + 1)$, the attachment of a square shape amounts to adding $(M + 1)$ columns, as required.

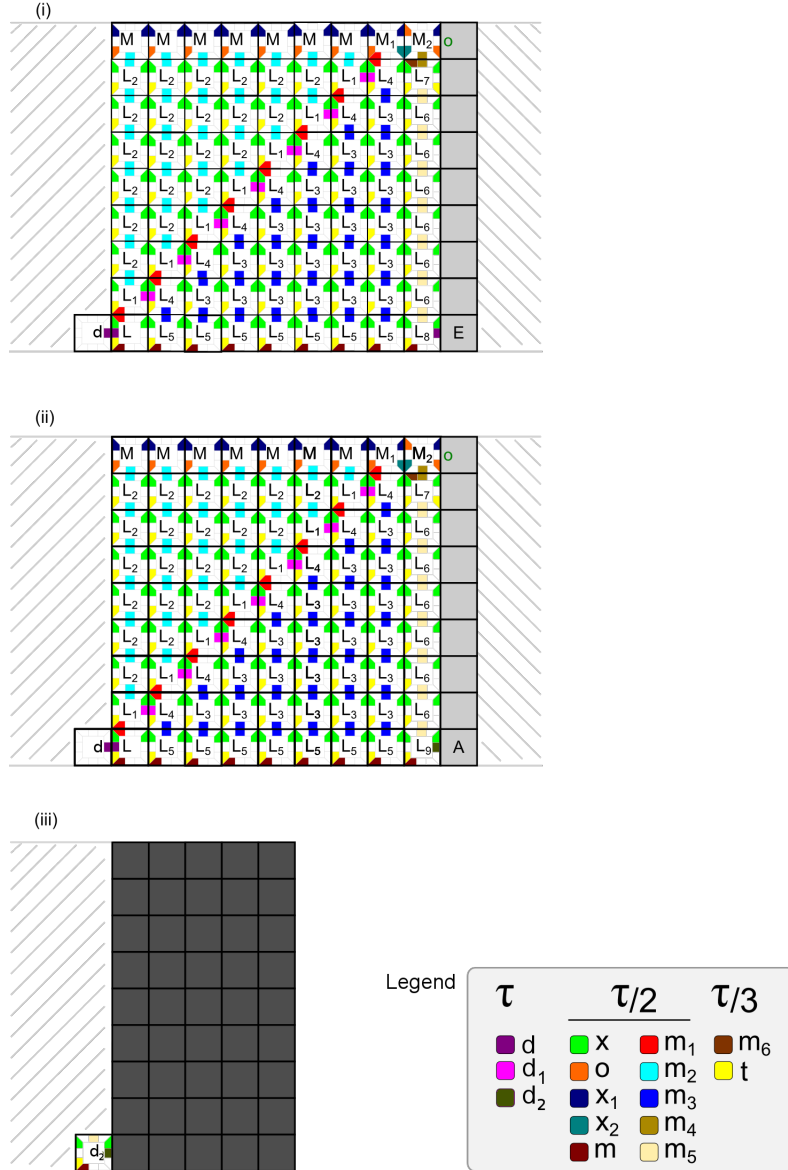


Figure 4.14: Together with the previous figure (Figure 4.13), this figure illustrates Step 3 (expansion), and Step 4 (reattachment). (i) The column underneath tile M_2 is completed, using tiles L_7, L_6 and L_8 . This refers to the case when the bottom left tile of the right part of the rectangle is tile E , which signifies that there are still columns to be expanded. (ii) illustrates the case when the bottom left tile of the right part of the rectangle is tile A , which signifies the end of the expansion process. In this case, tile L_9 attaches instead of tile L_8 as the last tile of the column headed by M_2 , and the process is continued by the construction of a new dark grey border of the rectangle, as illustrated in (iii).

The construction of this square is shown in Figure 4.13 and 4.14. Starting from tile L , the square self-assembles column by column, using the tile types and glues shown in Figure 4.13 (i). The last column of the square has tile M_2 in the top row. Tile M_2 and its mirror tile in

the tile set Θ' bring the corresponding left and right parts of the rectangles together via the glue o (Figure 4.13 part (ii)). Tiles L_7 , L_6 , and L_8 complete this last column, filling the gaps between the corresponding left and right parts of the rectangles (see Figure 4.14 part (i)), and thus reattaching the left and right parts of the rectangles to each other.

Step 5 - Initiate the repetition of Steps 2-4 above.

The last tile of the square constructed in Step 3, tile L_8 , sends a signal to the tile situated on its east side, which initiates Step 5 - the repetition of Steps 2-4.

Note that the replacement of a column with a square in Steps 3 & 4 is different for the very last column. In this case, the last tile of the square is, instead of tile L_8 , a special tile L_9 . This tile could not attach previously, because tile E , its east neighbour, does not attach to it, and can only attach when all tiles E have been consumed, and only tile A remains on the same row (see Figure 4.9 (iv), Figure 4.14(ii)). Tile A attaches to tile L_9 , which sends a signal to tile A which travels through the dark grey part and detaches all of it. Instead of starting a square construction, tile L_9 starts to build a new dark grey portion and the two rows underneath it (see Figure 4.9(v), (vi) and Figure 4.14(iii)). The tiles that are required to build this new “border” are hard-coded, and are not included in the tile set presented in Figure 4.11. The construction of the new border needs $O(C)$ tiles, and the construction of both dark grey borders is the only part of the construction that needs more than a constant number of tiles.

As a result of Steps 1-6, all $M!$ columns, except the dark grey columns, are replaced by $(M+1) \times (M+1)$ squares. Thus, the new rectangle has width $M! \times (M+1) + 2C = (M+1)! + 2C$.

The procedure above, which started with a rectangle of size $6 \times (6! + 2C)$, where $C = \lceil \log(N - 5) + 2 \rceil$, is then repeated $N - 6$ times.

To initiate the termination of the self-assembly, the left dark grey border is used. The left dark grey border acts as a counter, as follows. In order to count the number of rows and stop at N , the tiles F_1 , F_2 , F_3 and F_4 are used in the standard way, see, e.g., [15][3].

When the counter tile reaches N and stops counting, tile G' attaches to the leftmost column (instead of tile G), at the beginning of Step 1, see Figure 4.9(iii). Tile G' does not pass the column-detachment signal from its south edge to its east edge and instead will stop the growth

process, as follows. Tile G' initiates a signal to deactivate all the glues on the south edges of the last row. As a result, the rectangle above this last row detaches, and cannot grow anymore. This rectangle has size $N \times N!$.

The complete list of tile types in Figure 4.11, together with their signals can be found in Table 4.1 and Table 4.2. Note that the signal s_t , not shown in any of the tiles, starts from tile G' - which replaces tile G in Figure 4.9(iii) during the last application of Step 6. It then travels towards the east of tile G' : If a tile has an active glue s_t on its west edge, this signal activates s_t on its east edge. In addition, for all tiles it encounters, s_t deactivates all glues from the south edge, and thus detaches the mirror rectangle from the output rectangle. The signal s_t is common to all tiles, and is not shown in this table, for readability reasons. Signal s_1 is activated by tile C ; the path that it travels is shown in green in Figure 4.9(iv). Once it reaches tile B , signal s_1 is converted to signal s_5 . Tile E changes signal s_5 to s_6 (shown in black, respectively orange in Figure 4.12(i)), and s_6 starts the column detachment process. Signals s_7 , s_8 (red), and s_9 (green), s_{10} (blue) in Figure 4.12(iii) and Figure 4.12(vii), are all used in column detachment, as follows. Signals s_7 and s_8 are used to travel to the tiles above L . Signal s_9 deactivates the west glues of a column tile, while signal s_{10} deactivates the west edge of its neighbouring east tile, which belongs to the right part of the rectangle. Similarly, signal s_2 is the signal activated by the tile that mirrors C ; the path it travels is shown in blue in Figure 4.9(iv), and it will lead to the column detachment in the mirror rectangle.

Note that, since all self-assemblies happen in parallel, it would in principle be possible for various rectangle components to assemble at the same time, bringing the potential of undesired attachments. During the entire construction of the thin rectangle, the existence of the mirror rectangle prevents the attachment of incorrect left and right parts of rectangles (with different number of rows, or erroneous number of columns), by posing a geometrical constraint.

More precisely, in Step 5 (reattachment) the left and the right parts of the corresponding rectangle(s) must reattach, and this is where the possibility of an incorrect attachment occurs. Since the reattachment of the two parts happens through tile M_2 and its mirror tile in Θ' (marked in red in Figure 4.15), and since these two tiles only appear in the top row of the left part (the bottom row of the right part), the vertical distance between these two tiles in a potential attachment guarantees that the two parts of the rectangle(s) have the same number of rows.

Tile Name	List of Signals				Transitions
	North	West	South	East	
<i>A</i>		s_2^-		s_2^-	$(s_2, W) \rightarrow \{(s_2, E)\}$
<i>B</i>			s_1^-, s_2^-	s_5^-, s_2^-	$(s_2, W) \rightarrow \{(s_2, E)\},$ $(s_1, S) \rightarrow \{(s_5, E)\}$
<i>C</i>			s_2^-	s_1^+, s_2^-	$(s_2, E) \rightarrow \{(s_2, S)\}$
<i>E</i>		$s_2^-, s_6^-,$ s_5^-		s_2^-, s_6^-	$(s_2, W) \rightarrow \{(s_2, E)\},$ $(s_5, W) \rightarrow \{(s_6, E), (e, W), (s_1, E)\},$ $(s_6, W) \rightarrow \{(e, W)\}$
<i>F₀</i>		s_2^-		s_2^-	$(s_2, W) \rightarrow \{(s_2, E)\}$
<i>F₁</i>		s_2^-		s_2^-	$(s_2, W) \rightarrow \{(s_2, E)\}$
<i>F₂</i>		s_2^-		s_2^-	$(s_2, W) \rightarrow \{(s_2, E)\}$
<i>F₃</i>		s_2^-		s_2^-	$(s_2, W) \rightarrow \{(s_2, E)\}$
<i>F₄</i>		s_2^-		s_2^-	$(s_2, W) \rightarrow \{(s_2, E)\}$
<i>G</i>			s_2^-	s_2^-	$(s_2, S) \rightarrow \{(s_2, E)\}$
<i>G'</i>					s_t^+
<i>H</i>		s_2^-		s_2^-	$(s_2, W) \rightarrow \{(s_2, E)\}$
<i>I</i>		s_2^-	s_2^-		$(s_2, W) \rightarrow \{(s_2, S)\}$
<i>J</i>		s_1^-, s_2^-		s_1^-, s_2^-	$(s_1, W) \rightarrow \{(s_1, E)\}$ $(s_2, E) \rightarrow \{(s_2, W)\}$
<i>K</i>	s_2^-	s_1^-, s_2^-	s_1^-		$(s_1, W) \rightarrow \{(s_1, S)\}$ $(s_2, N) \rightarrow \{(s_2, W)\}$
<i>L</i>	s_8^-, s_7^+ s_9^-	s_5^-	$s_7^-, s_8^-,$ s_9^-	s_{10}^-	$(s_5, W) \rightarrow \{(s_7, N)\},$ $(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_9, N) \rightarrow \{(t, W), (x, W), (d, W)\}$

Tile Name	List of Signals				Transitions
	North	West	South	East	
L_1	s_8^-, s_9^-, s_{10}^-	s_{10}^-	$s_7^-, s_8^-, s_9^-, s_{10}^-$	s_{10}^-	$(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_9, N) \rightarrow \{(t, W), (x, W)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\},$ $(s_{10}, N) \rightarrow \{(t, W), (x, W)\}$
L_2	s_8^-, s_9^-, s_{10}^-	s_{10}^-	$s_7^-, s_8^-, s_9^-, s_{10}^-$	s_{10}^-	$(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_9, N) \rightarrow \{(t, W), (x, W)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\},$ $(s_{10}, N) \rightarrow \{(t, W), (x, W)\}$
L_3	s_8^-, s_9^-, s_{10}^-	s_{10}^-	$s_7^-, s_8^-, s_9^-, s_{10}^-$	s_{10}^-	$(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_9, N) \rightarrow \{(t, W), (x, W)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\},$ $(s_{10}, N) \rightarrow \{(t, W), (x, W)\}$
L_4	s_8^-, s_9^-, s_{10}^-	s_{10}^-	$s_7^-, s_8^-, s_9^-, s_{10}^-$	s_{10}^-	$(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\},$ $(s_{10}, N) \rightarrow \{(t, W), (x, W)\},$ (d_1, W)

Table 4.1: Together with Table 4.2, this table list the signals and transitions for all the tiles types, except the dark grey border areas. Signals are denoted by the letter s , with subscripts or superscripts, while the other letters indicate glues. The signal s_t , not shown in any of the tiles, starts from tile G' - which replaces tile G in Figure 4.9(iii) during the last application of Step 6. It then travels towards the east of tile G' : If a tile has an active glue s_t on its west edge, this signal activates s_t on its east edge. In addition, for all tiles it encounters, s_t deactivates all glues from the south edge, and thus detaches the mirror rectangle from the output rectangle. The signal s_t is common to all tiles, and is not shown in these tables, for readability reasons. Signal s_1 is activated by tile C ; the path that it travels is shown in green in Figure 4.9(iv). Once it reaches tile B , signal s_1 is converted to signal s_5 . Tile E changes signal s_5 to s_6 (shown in black, respectively orange in Figure 4.12(i)), and s_6 starts the column detachment process. Signals s_7, s_8 (red), and s_9 (green), s_{10} (blue) in Figure 4.12(iii) and Figure 4.12(vii) are all used in column detachment, as follows. Signals s_7 and s_8 are used to travel the tiles above L . Signal s_9 deactivates the west glues of a column tile, while signal s_{10} deactivates the west glues of its neighbouring east tile, which belongs to the right part of the rectangle. Similarly, signal s_2 is the signal activated by the tile that mirrors C ; the path it travels is shown in blue in Figure 4.9(iv), and it will lead to the column detachment in the mirror rectangle.

Tile Name	List of Signals				Transitions
	North	West	South	East	
L_5	s_8^-, s_9^-, s_{10}^-	s_{10}^-	$s_7^-, s_8^-, s_9^-, s_{10}^-$	s_{10}^-	$(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_9, N) \rightarrow \{(t, W), (x, W)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\},$ $(s_{10}, N) \rightarrow \{(t, W), (x, W)\}$
L_6	s_8^-, s_9^-, s_{10}^-	s_{10}^-	$s_7^-, s_8^-, s_9^-, s_{10}^-$	s_{10}^-	$(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_9, N) \rightarrow \{(t, W), (x, W)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\},$ $(s_{10}, N) \rightarrow \{(t, W), (x, W)\}$
L_7	s_8^-, s_9^-, s_{10}^-	s_{10}^-	$s_7^-, s_8^-, s_9^-, s_{10}^-$	s_{10}^-	$(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_9, N) \rightarrow \{(t, W), (x, W)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\},$ $(s_{10}, N) \rightarrow \{(t, W), (x, W)\}$
L_8	s_8^-, s_9^-	s_{10}^-	$s_7^-, s_8^-, s_9^-, s_{10}^-$	s_{10}^-	$(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_9, N) \rightarrow \{(t, W), (x, W)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\},$ $(s_{10}, N) \rightarrow \{(t, W), (x, W)\}$
L_9	s_8^-, s_9^-, s_{10}^-	s_{10}^-	$s_7^-, s_8^-, s_9^-, s_{10}^-$	s_{10}^-	$(s_7, S) \rightarrow \{(s_8, N)\},$ $(s_8, S) \rightarrow \{(s_9, S), (s_{10}, E)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\},$ $(s_{10}, N) \rightarrow \{(t, W), (x, W)\}$
M		s_{10}^-	s_{10}^-		$(s_{10}, W) \rightarrow \{(s_{10}, S)\}$
M_1		s_{10}^-	s_{10}^-		$(s_{10}, W) \rightarrow \{(s_{10}, S)\}$
M_2		s_{10}^-	s_4^-, s_{10}^-		$(s_4, S) \rightarrow \{(o, W), (x_2, W),$ $(m_4, S), (x_1, E), (o, E)\},$ $(s_{10}, W) \rightarrow \{(s_{10}, S)\}$
M_3		s_{10}^-	s_4^+, s_{10}^-		$(s_{10}, W) \rightarrow \{(s_{10}, S)\}$

Table 4.2: Together with Table 4.1, this table list the signals and transitions for all the tiles types, except the dark grey border areas. See caption of Table 4.1 for details.

The horizontal distance between these two tiles in a potential attachment guarantees that the two parts of the rectangle(s) have the same number of columns in the interlocking portion. Together, these two conditions guarantee that the left and right parts of the rectangle(s) belong to the same step.

Note that tile G , which marks the final configuration, can only attach to the leftmost column when the assembled rectangle has size $N \times N!$. As a result, the proposed DTAM tile assembly system only assembles rectangles of size $N \times N!$.

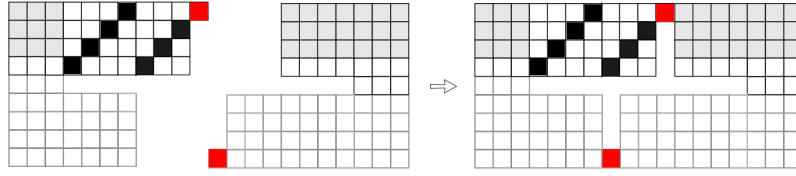


Figure 4.15: The mirror rectangle guarantees correct reattachments in Step 5, through the geometrical constraints which it imposes on the attachment. The tiles marked in red are the only tiles through which the left and right parts of the rectangle(s) can reattach in Step 5.

This concludes the construction and the proof of Theorem 4.4.1.

4.4.3 DTAM tile complexity and waste analysis

The tile sets Θ and Θ' that are used to build the rectangles and mirror rectangles have a constant number of tiles. In order to construct an $N \times N!$ rectangle, we start from a small rectangle of size $6 \times (6! + 2\lceil \log(N - 5) + 2 \rceil)$. Assuming that the assembly of this initial rectangle needs $O(\log(N))$ tile types, the entire construction will need a constant number of tile types for Steps 2-4, and $O(\log(N))$ tile types to construct the dark grey borders, totalling $O(\log(N))$ tiles types.

In [1], tile complexities of various constructions of “thin rectangles” are compared. The construction of an $N \times N!$ rectangle using the most economic self-assembly models (multi-temperature self-assembly model, q -tile model, or unique shape model [1]) have a tile complexity of $O(\log N! / \log \log N!)$, that is, use approximately $O(N)$ tile types. In comparison, our DTAM self-assembly system utilizes only $O(\log N)$ tile types.

Regarding waste analysis, note first that the waste configurations that are produced during the assembly process cannot reattach to the non-waste main or mirror rectangle. In addition, the waste configurations cannot attach to single tiles. Lastly, the waste configurations cannot

attach to other waste configurations.

As a result, and since during each repeat of Step 1 through Step 5 all light grey columns of the rectangle in Figure 4.7 are replaced with new squares, the total size of the waste supertiles that is generated to build an $N \times N!$ rectangle is $O(6 \times 6!) + O(7 \times 7!) + \dots + O((N-1) \times (N-1)!)$ which is smaller than the size of the final configuration $O(N \times N!)$.

4.5 Conclusions

This paper introduces a simplified version of the Signal Tile Assembly Model (STAM), called DTAM (Detachable Tile Assembly Model) which is based on the 2-HAM and uses a restricted version of signals, namely signals that can only deactivate glues (as opposed to the glue-activating and glue-deactivating signals of STAM). We show that a doubly-simplified version of STAM, namely SDTAM, that uses glue-deactivating signals only and, in addition, uses only one-tile-at-a-time attachments, is still capable of universal Turing computation at temperature one. Moreover, our Turing-simulating SDTAM uses at most one signal per tile, and signals travel through only one tile before deactivating a glue. All these simplifications, that are achieved without sacrificing any computational power, could have potential implications on the practical implementations of signal-based tile assembly systems. We also present a DTAM that assembles a thin $N \times N!$ rectangle, and has a lower tile-complexity than existing constructions. This illustrates the potential benefits of the DTAM model for tile-complexity reduction.

Bibliography

- [1] G. Aggarwal, Q. Cheng, M. Goldwasser, M.-Y. Kao, P. M. de Espanés, and R. Schweller. Complexities for generalized models of self-assembly. *SIAM J. Comput.*, 34(6):1493–1515, 2005.
- [2] B. Behsaz, J. Manuch, and L. Stacho. Turing universality of step-wise and stage assembly at temperature 1. In *DNA Computing and Molecular Programming - International Conference, DNA 18. Proceedings*, volume 7433 of *LNCS*, pages 1–11, 2012.

- [3] H.-L. Chen, R. Schulman, A. Goel, and E. Winfree. Reducing facet nucleation during algorithmic self-assembly. *Nano Letters*, 7(9):2913–2919, 2007.
- [4] M. Cook, Y. Fu, and R. Schweller. Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. In *SODA*, pages 570–589. SIAM, 2011.
- [5] E. Demaine, M. Demaine, S. Fekete, M. Ishaque, E. Rafalin, R. Schweller, and D. Souvaine. Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues. *Natural Computing*, 7(3):347–370, 2008.
- [6] D. Doty, L. Kari, and B. Masson. Negative interactions in irreversible self-assembly. *Algorithmica*, 66(1):153–172, 2013.
- [7] T. Fochtman, J. Hendricks, J. Padilla, M. Patitz, and T. Rogers. Signal transmission across tile assemblies: 3D static tiles simulate active self-assembly by 2D signal-passing tiles. *Natural Computing*, 14(2):251–264, 2015.
- [8] J. Hendricks, M. Patitz, and T. Rogers. Reflections on tiles (in self-assembly). In A. Phillips and P. Yin, editors, *DNA Computing and Molecular Programming - International Conference, DNA 21. Proceedings*, volume 9211 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 2015.
- [9] A. Keenan, R. Schweller, and X. Zhong. Exponential replication of patterns in the signal tile assembly model. In *DNA Computing and Molecular Programming - International Conference, DNA 19. Proceedings*, volume 8141 of *LNCS*, pages 118–132, 2013.
- [10] J. Padilla, W. Liu, and N. Seeman. Hierarchical self assembly of patterns from the Robinson tilings: DNA tile design in an enhanced Tile Assembly Model. *Natural Computing*, 11:323–338, 2012.
- [11] J. Padilla, M. Patitz, R. Pena, R. Schweller, N. Seeman, R. Sheline, S. Summers, and X. Zhong. Asynchronous signal passing for tile self-assembly: Fuel efficient computation and efficient assembly of shapes. In *UCNC 2013*, volume 7956 of *LNCS*, pages 174–185, 2013.

- [12] M. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014.
- [13] M. Patitz, R. Schweller, and S. Summers. Exact shapes and Turing universality at temperature 1 with a single negative glue. In L. Cardelli and W. M. Shih, editors, *DNA Computing and Molecular Programming - International Conference, DNA 17*, volume 6937 of *Lecture Notes in Computer Science*, pages 175–189. Springer, 2011.
- [14] J. Reif, S. Sahu, and P. Yin. Complexity of graph self-assembly in accretive systems and self-destructible systems. *Theor. Comput. Sci.*, 412(17):1592–1605, 2011.
- [15] P. Rothemund and E. Winfree. The program-size complexity of self-assembled squares (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 459–468, 2000.
- [16] N. Seeman. Nucleic acid junctions and lattices. *Journal of Theoretical Biology*, 99(2):237 – 247, 1982.
- [17] E. Winfree. *Algorithmic self-assembly of DNA*. PhD thesis, 1998.
- [18] E. Winfree, F. Liu, L. Wenzler, and N. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394:539–544, 1998.
- [19] E. Winfree, X. Yang, and N. Seeman. Universal computation via self-assembly of DNA: some theory and experiments. In *DNA Based Computers 1996, Proceedings of a DIMACS Workshop*, pages 191–214, 1996.

4.6 Addendum

More Details on the Definition of Binding Graphs

A *pseudo-grid graph* is a directed weighted graph $G = (N_G, E_{G_v} \cup E_{G_h}, \pi_G)$ where N_G is a finite set of nodes, $E_{G_v}, E_{G_h} \subseteq N_G \times N_G$ are two sets of edges (called the vertical and horizontal edges respectively) such that $E_{G_v} \cap E_{G_h} = \emptyset$, and the function $\pi_G : (E_{G_v} \cup E_{G_h}) \rightarrow \mathbb{Z}^+$ is a weight function that associates a weight to every edge in the graph. The weight function assigns a number to each edge of the graph. In the context of the associated binding graph to a supertile, the weight of the edge between two neighbour tiles t_1 and t_2 is the $\sigma(g)$ where g is the common glue on the common edge of t_1 and t_2 , and σ is the glue function.

A *cut* on the graph G is an unordered pair of set of nodes $C = \{N_1, N_2\}$, where $N_1 \subset N_G$, $N_2 \subset N_G$, $N_1 \cup N_2 = N_G$ and $N_1 \cap N_2 = \emptyset$. The weight of the cut C is defined to be the sum of the weight of all the edges that connect a node from N_1 to a node from N_2 , or a node from N_2 to a node from N_1 .

A configuration consists of only one tile if and only if its associated binding graph has only one node.

Corrections

Page 71-72. t_f should be replaced with f in the last paragraph of page 71, and first paragraph of page 72. The correct text is as follows.

A *Detachable Tile Assembly System* based on *Detachable Tile Assembly Model (DTAM)* over alphabets Γ and Σ is a 5-tuple $(\Theta, g, \tau, \lambda, f)$, where Θ is a set of tiles over $\Gamma \times \Sigma$, g is a strength function over Γ , τ is a positive integer (temperature), λ is a finite set of starting τ -stable configurations, and $f \in \Theta$ is a single tile which will “mark” the final configuration of the assembly. The self-assembly process begins with the set of configurations $\lambda \cup \{\text{all configurations that consist of one tile}\}$ and proceeds by successive applications of the three types of transitions to either one configuration (in the case of a detachment or an action transition) or two configurations (in the case of an attachment transition) in the current set of configurations, asynchronously and non-deterministically. After applying each transition, the resulting

configurations are added to the current set of configurations. We say that a set of configurations C is *derived from* $(\Theta, g, \tau, \lambda, f)$ if, starting from $\lambda \cup \{\text{all configurations that consist of one tile}\}$, we can obtain C by iteratively applying any of the three types of transitions. We call a supertile Z a *final supertile* if there exists a set of configurations C derived from $(\Theta, g, \tau, \lambda, f)$ with the property that $Z \in C$ and Z is a τ -stable signal-stable supertile such that there exist $i, j \in \mathbb{Z}^2$ with $Z(i, j) = f$.

Chapter 5

Smart Tile Self-Assembly and Replication¹

5.1 Introduction

Self-assembly in nature is one the main inspirations for multi-robot systems [14, 21]. In such multi-robot systems, robots can cooperate in order to move blocks and build structures [22], assemble complex two-dimensional shapes [13, 19], or replicate given two-dimensional shapes [8]. Robots in many multi-robot systems do not use any central control, and use instead their limited local computational capabilities and communication with nearby robots to accomplish these tasks. Indeed, robots in multi-robot systems often do not need to have extensive computational power [7].

In the context of DNA computing and molecular programming, self-assembly of DNA tiles has been used extensively for either computational tasks or for the assembly of complex nanostructures [17]. Traditionally, these tiles have been “static” in the sense that they are unchangeable entities that can attach to their neighbouring tiles based on pre-programmed local attachment rules. However, recently, an “active” dimension has been added to tiles in the form of, e.g., the addition of signal transmission that can activate or deactivate glues on the tile edges [15, 16, 10]. On the other hand, several DNA-based computational devices have been

¹This chapter is based on the paper: Lila Kari, Amirhossein Simjourn: Smart Tile Self-Assembly and Replication. *Fundam. Inform.* 154(1-4): 239-260 (2017)

implemented, such as finite state automata [2], computational state transitions [12], and digital circuits [18], to name just a few.

In this paper, we bring together these three ideas, multi-robot systems with local computing capabilities, static DNA tiles, and DNA-based computational devices, by proposing a dynamic self-assembly model based on *smart tiles*, that is, dynamic tiles which are equipped with local computational devices in addition to being able to transmit signals that can activate or deactivate glues.

Smart tiles are based on the classical tiles as defined in the abstract tile self-assembly model, aTAM [23], but they are enhanced with the ability to transmit signals that can activate or deactivate glues, as proposed in the signal tile assembly model STAM [16, 15]. The novel feature of smart tiles, which we propose in this paper, is the fact that each is equipped with a local computational device.

We show the potential of smart tile self-assembly (smart-TAM) systems to perform complex tasks by proving their ability to replicate arbitrary convex shapes, in a manner that is similar to the 2D shape replication by smart sand robotic systems [8]. Note that, in contrast to previous methods for pattern replication by self-assembly [11], the smart tile self-assembly systems we use for shape replication do not make any special assumptions about the tiles located in the interior of the shape. Moreover, in contrast to other methods for shape replication by self-assembly systems [4] [9] [1], our proposed smart tile assembly replication system is not in-place. This means that our construction assembles a replica separately from the original object, rather than cutting out the original object to leave an imprint to be filled in by the replica. We also note that [4] uses negative glues and is based on the 2HAM attachment model, wherein two supertiles can attach to each other during an attachment transition, while our model allows attachment of one tile only to the existing structure, during each transition. Another replication system, [1], is based on the staged tile assembly model, which allows changes of the tile set at different stages of the assembly process.

Overall, the model we propose offers a general framework to discuss and compare various tile self-assembly systems. For example, STAM with an one-tile-at-a-time attachment model becomes a particular case of smart-TAM wherein the local tile computational device is a simple look-up table, and activation of deactivated glues is not allowed. The local tile computational

device that we use in the smart-TAM system presented in this paper, for two-dimensional shape replication, is a generalized sequential machine (a deterministic finite state automaton with output). At the other end of the spectrum, the local tile computational device can be as complex as a Turing Machine.

The paper is organized as follows. Section 5.2 introduces basic definitions and notations, as well as formally defines a smart tile assembly system. Section 5.3 constructs a smart-TAM system that can replicate a supertile of any L -convex shape, and indicates how this construction can be easily modified to replicate any convex shape. Section 5.4 discusses complexity issues and illustrates the idea for a construction of a smart-TAM system with lower tile and space complexity, which could replicate a supertile of any shape.

5.2 Smart Tile Assembly

In this section we will introduce the Smart Tile Assembly Model (smart-TAM), a tile self-assembly model with tiles that are equipped with identical local computational devices. The local computational device may be as simple as a look-up table or as complex as a Turing machine. The only constraint placed on the computational devices is that the input and the output must be defined based on the signals and the glues on the edges of the tile.

5.2.1 Basic Definitions

The following basic definitions will be used in the formal description of the smart tile assembly model.

Given a set A , its cardinality is denoted by $|A|$, and the set of all subsets of A is denoted by $\mathcal{P}(A)$. A *multiset* is a generalized set, the elements of which can appear more than once. The multiplicity of an element a from a multiset S is the number of times that a appears in S . The cardinality of a multiset is the sum of the multiplicities of its elements. For example, if $S = \{a, a, b, c, d, c\}$ is a multiset, the multiplicity of a in S is 2 and the cardinality of S is 6.

The set of *directions* is defined as $D = \{N, E, S, W\}$, and the elements in D represent the directions north, east, south and west respectively.

In this paper we will define and investigate smart tile assembly systems, which use smart tiles, that is, tiles endowed with a local computational device. The local computational device that we will use to illustrate the capabilities of such self-assembly systems is the generalized sequential machine. A *generalized sequential machine* (GSM) is similar to a finite nondeterministic automaton but which can output a word for each input letter it reads. Formally, a GSM is a sextuple $g = (S, V_I, V_O, s_0, S_f, P)$ where S is the set of *states*, V_I is the *input* alphabet, V_O is the *output* alphabet, with $S \cap (V_I \cup V_O) = \emptyset$, and s_0 is an element in S called the *start state*, $S_f \subseteq S$ is the set of *final states*, and the *productions* in P are of the form

$$s_i a \longrightarrow w s_j, \quad s_i, s_j \in S, \quad a \in V_I, \quad \text{and } w \in V_O^*.$$

For a generalized sequential machine g and a word u over its input alphabet V_I , we denote

$$g(u) = \{w | s_0 u \Longrightarrow^* w s_1, \text{ for some } s_1 \in S_f\}$$

where \Longrightarrow^* is derivation relation induced by \longrightarrow . If L is a language over V_I , then the GSM *translates*, or *maps*, L into the language

$$g(L) = \{w | w \in g(u) \text{ for some } u \in L\}.$$

A generalized sequential machine is deterministic iff, for every $s_i \in S$ and $a \in V_I$, there is exactly one production in P . A deterministic GSM is a *Mealy machine* iff all words w appearing in productions in P consists of only one letter in V_O .

The smart tile assembly model that we will define is a generalization of the *Signal Tile Assembly Model (STAM)*, which was introduced by Padilla et al. [16], which itself is an extension of the Abstract Tile Assembly Model (aTAM) introduced by Winfree in [23]. More precisely, STAM is a tile assembly model based on 2HAM [5, 6], wherein each tile possesses a set of glues on each edge (instead of one glue per edge, like in aTAM and 2HAM), and glues can be activated or deactivated by signals. In addition, unlike aTAM, where the growth of an assembled structure happens one tile at a time, in STAM (as in 2HAM), a whole multi-tile structure that was assembled separately can attach to an existing structure in one attachment step.

In the STAM model, the status of each glue on an edge can be *latent*, *on*, or *off*. Only a glue whose status is *on* is active and can contribute to attaching the tile to another tile with an identical glue with *on* status on the abutting edge. If the status of a glue is *off* or *latent*, the glue is inactive and it does not have any attachment capabilities. In order to change the status of glues in STAM, signals are used. Intuitively, a signal is a mapping associated to a given tile that assigns to a glue on an edge a set of changes in the status of the glues on the other edges. For example, assume that tile t has glue g_e on its east edge, glue g_s on its south edge, and glue g_n on its north edge. Also assume that all these glues are *on*, and assume that there is a signal on the east side of the tile t that assigns a change of the status of the glue g_s to *off*. If that is the case, and if the tile t attaches to another tile via its east edge, the signal deactivates the glue g_s , that is, it changes its status to *off*. Signals can change the status of a glue from *latent* to *on* or to *off*, or from *on* to *off*. Note that, once a glue is in the *off* state, its status cannot be changed anymore. A tile can send a signal to its neighbour tile by activating a glue on the edge that they have in common. Signals can change the status of the glues, therefore signals can activate new glues and thus initiate a signal in the next tile. Moreover, signals can activate glues on a free (unattached) edge and make new attachments possible. In addition to the activation, signals can deactivate glues and, as a result, an existing structure might become unstable. In the STAM model, if the deactivation of a glue makes a structure unstable, the structure will break apart into stable components.

More formally, if Γ is a set of glues, Σ is a set of tile labels, $Q = \{on, off, latent\}$, and D is the set of directions, then an STAM tile over the alphabet $\Gamma \times \Sigma$ is a quadruple $t = (G, L, \Pi, \delta)$, where $G : D \rightarrow \mathcal{P}(\Gamma \times Q)$ denotes the sets of glues on the edges of t , and $L \subseteq \Sigma$ is a set of tile labels. The transition function $\delta : D \times \Gamma \rightarrow \mathcal{P}((D \times \Gamma \times \{on, off, latent\}) \cup (\Sigma \times \{on \times off\}))$ defines the set of actions that result as a consequence of an attachment through a glue on one of the edges of the tile t . More precisely, the transition function of the tile t associates to a glue on one of its edges changes of the status of glues on other edges, or a change of a tile label, or both. These outputs are called actions. During a self-assembly process, when a transition is applied, the outputs of the transition function are added to Π , the multiset of pending actions. For a detailed formal definition of STAM, the reader is referred to [16].

5.2.2 The Smart Tile Assembly Model

Informally, in the smart tile assembly model, a local computational device is added to each tile which, if it is in a certain state and receives as input a glue and a direction, it changes its state and outputs an action that amounts to the activation or deactivation of some other glues and tile labels. This is a generalization of the STAM feature wherein the mapping between the glues and the actions was defined by the *transition function*. In the smart tile assembly model, this role is played by the local computational device, which may be as simple as a look-up table or as complex as a Turing machine.

Smart tiles: A *smart tile* can be viewed as a unit square with a tile label, as well as glues on each of its four edges. A *glue* over the glue alphabet Γ is a triplet (γ, d, q) where $\gamma \in \Gamma$, the alphabet of glues, $d \in D$ is one of the directions, and $q \in \{on, off\}$. A *tile label* over the tile label alphabet Σ is a pair (σ, q) where $\sigma \in \Sigma$ and $q \in \{on, off\}$. Moreover, a smart tile is endowed with a computational device C , that can change the status q of glues and tile labels.

Formally, a *smart tile* over the alphabet $\Gamma \times \Sigma$ is a quadruple (G, L, Π, C) that has a set of glues G over the glue alphabet Γ , a set of labels L over the tile label alphabet Σ , a multiset of a *pending actions* Π with elements from $(\Gamma \times D \times \{on, off\}) \cup (\Sigma \times \{on, off\})$, and a *tile computational device* C . A computational device C in the context of smart-tile assembly is a rewriting system with input alphabet $I \subseteq \Gamma \times D$, and output alphabet $O \subseteq \mathcal{P}((\Gamma \times D \times \{on, off\}) \cup (\Sigma \times \{on, off\}))$. Generalized sequential machines (finite automata with output) and Turing machines are examples of valid computational devices.

Definition A *smart tile assembly system (smart-TAM system)*, over the alphabet $\Gamma \times \Sigma$ is a quadruple $(\Theta, g, \tau, s_{seed})$, where Θ is a set of smart tiles over the same alphabet, $g : \Gamma \rightarrow \mathbb{N}$ is the *glue-strength function* that defines the binding strength of each glue, $\tau \in \mathbb{N}$ is the *temperature* which defines the minimum total strength required for an attachment of a tile to occur, and $s_{seed} \in \Theta$ is the *seed* smart tile, which is the tile from which the self-assembly process starts.

A *configuration* over the set of smart tiles Θ is a mapping $c : \mathbb{Z}^2 \rightarrow \Theta$. Informally, a configuration is a placement of tiles on the rectangular $\mathbb{Z} \times \mathbb{Z}$ grid, with the centers of the tiles placed on the integer coordinate nodes of the grid. If for a position (x, y) we have that $c(x, y)$ is undefined, we will say that $c(x, y) = \text{null}$. Two smart tiles in a configuration are called *adjacent*

if the Euclidean distance between their centers is equal to 1. In other words, two smart tiles are adjacent if they have a common edge.

The weighted graph $B = (N, E, W)$ with the set of nodes N , set of edges E and the weight function $W : E \rightarrow \mathbb{N}$ is the binding graph of a configuration $c : \mathbb{Z}^2 \rightarrow \Theta$ if there exists a bijective mapping between the set of nodes N and $\text{dom}(c)$, such that two nodes in N are connected by an edge in the graph if and only if the corresponding smart tiles in the configuration are adjacent and, moreover, the strength of the attachment between the tiles equals the weight of the edge connecting the corresponding nodes in the graph. A configuration is called a *supertile* iff its binding graph is connected. A weighted graph G has a *cut* with weight w if one can partition the nodes of G into two disjoint sets such as the sum of the weights of the edges between these two sets is w . A supertile is called τ -stable if its binding graph does not have any *cut* with weight less than τ . If its binding graph G has at least one cut with the weight $w < \tau$, then the supertile is called τ -unstable.

The self-assembly in a smart-TAM system proceeds through transitions: A supertile V is obtained from the supertile U in one transition, iff the supertile V is the result of one of the following three types of transitions applied to the supertile U : attachment transitions, signal transitions, and detachment transitions.

Attachment transition: The τ -stable supertile V is the result of the attachment of a new smart tile t to the τ -stable supertile U , in position (x, y) , iff $U(x, y) = \text{null}$, $V(x, y) = t$, and all other smart tiles in position (i, j) in U are the same as the tiles in position (i, j) in V , with the exception of the tiles in positions adjacent to (x, y) , where the following hold:

- The sum of the strengths of the glues on the common edges between tile t and all its adjacent tiles t' from U is greater than or equal to the temperature τ . In addition, for all such t' , the glues on the edges common with the newly attached smart tile t become inputs for the computational devices in both t and t' .
- For each of the inputs (on all newly attached edges) to the computational device of t and all its adjacent t' in U , one computational step is performed, and the output is added to

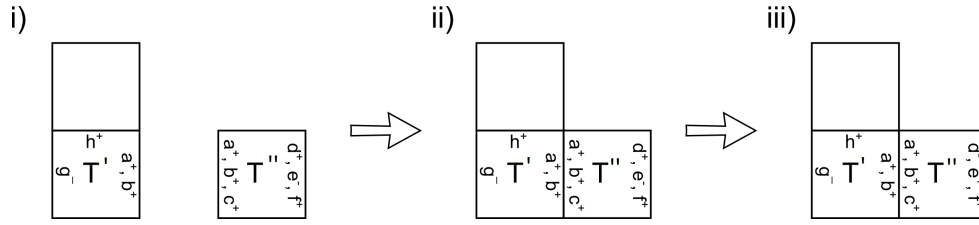


Figure 5.1: The attachment transition from Example 1. Note that superscripts are used to denote the status of glues and labels, with “+” indicating *on*, and “-” indicating *off*. (i) A smart supertile and a smart tile that can attach to it. (ii) The result of an attachment transition of the smart tile labelled T'' to the smart supertile containing the tile labelled T' . The glues of T'' after the attachment are changed due to the actions of the computational device on T'' . The pending actions that are added to the set of pending actions of T'' are $\{(d, E, \text{off}), (e, E, \text{on})\}$. (iii) The outcome of a signal transition applied to the smart tile T'' with pending actions $\{(d, E, \text{off}), (e, E, \text{on})\}$, in the supertile from (ii).

the multiset of pending actions of that smart tile. In case there are multiple inputs, the order in which they are processed is non-deterministic.

Example 1. The smart tile (G_1, L_1, Π_1, C_1) with tile label T' in Figure 5.1 is defined such that $G_1 = \{(a, E, \text{on}), (b, E, \text{on}), (g, W, \text{off}), (h, N, \text{on})\}$, $L_1 = \{T'\}$, $\Pi_1 = \emptyset$, and the computational device C_1 is defined so that it never changes its state and outputs the empty set for every input. The smart tile (G_2, L_2, Π_2, C_2) with tile label T'' is defined such that $G_2 = \{(a, W, \text{on}), (b, W, \text{on}), (c, W, \text{on}), (d, E, \text{on}), (e, E, \text{off}), (f, E, \text{on})\}$, $L_2 = \{T''\}$, $\Pi_2 = \emptyset$, and the computational device C_2 is the GSM with set of states $\{s_1, s_2\}$, the input alphabet $\{a, b, c, d, e, f, g, h\} \times D$, the output alphabet $\{(d, E, \text{off}), (e, E, \text{on})\}$, and transition function $(s_1, (a, E)) \longrightarrow \{s_2\} \times \{(d, E, \text{off}), (e, E, \text{on})\}$.

Figure 5.1(i) shows the smart tile T'' in the process of attaching to a a supertile containing the smart tile T' . Assume that the temperature of the system is $\tau = 2$ and that the strength of the glues a and b is equal to 1 (therefore the smart tiles T' and T'' attach via the glues a and b on the west edge of T'').

As a result of an attachment transition, the supertile in Figure 5.1(ii) is formed. During this attachment transition, the multiset Π_2 of the smart tile T'' will change from the empty set to $\Pi_2 = \{(d, E, \text{off}), (e, E, \text{on})\}$. Note that the attachment of the smart tiles T' and T'' changes the state of the computational device C_2 in T'' to s_2 , and generates the output $\{(d, E, \text{off}), (e, E, \text{on})\}$, which will be added to the set of pending actions which becomes $\Pi_2 = \{(d, E, \text{off}), (e, E, \text{on})\}$.

Note also that this output only changes the set of pending actions Π_2 and it does not have any effect on the status of the glues. The status of the glues will be changed during a subsequent *signal transition*.

Signal transition: The supertile V is the result of *signal transition* in the smart tile $t = (G, L, \Pi, C)$ from the position (x, y) of the τ -stable supertile U , iff t has at least one pending action in the set of its pending actions Π (activate or deactivate a glue or a label) and the following conditions hold. For all $(i, j) \in \mathbb{Z}^2$ all the smart tiles in the positions (i, j) in V are the same as the smart tiles in the positions (i, j) in U , except the smart tile t and the smart tiles t' that are adjacent to t which are changed as follows:

- The smart tile t removes one of the pending actions from its multiset Π and applies it to the indicated glue or tile label. The application of an action means changing the status of the corresponding glue or tile label accordingly.
- In addition, if the pending action is to activate a glue (change its status to *on*), then the computational device of the smart tile t' that is adjacent to t , on the same edge as the glue, performs a computational step with this glue as the input, and adds the output of this computational step to the set of its own pending actions. Subsequently, the used pending action is removed from the set Π of the pending actions of t .

Figure 5.1(iii) shows the outcome of the two signal transitions that result from applying the two pending actions $\{(d, E, \text{off}), (e, E, \text{on})\}$ of the smart tile T'' from Figure 5.1(ii).

Detachment transition: If a supertile U consists of two parts whose connection strength is lower than the temperature, then U can break into two supertiles. Formally, supertiles U_1 and U_2 are the result of a detachment transition of a τ -unstable supertile U if the following conditions hold: $G_V = (N, E, W)$ is the associated assembly-graph of the supertile U , and there exist two disjoint sets $N_1, N_2 \subset N, N_1 \cup N_2 = N$ such that the weight of the cut (N_1, N_2) is smaller than τ , and $G|_{N_1}$ is the assembly-graph associated to U_1 and $G|_{N_2}$ is the assembly-graph associated to U_2 .

A computation in a smart-TAM system starts from the seed smart tile and proceeds by non-deterministic applications of attachment transitions, signal transitions, and detachment transitions. A configuration is called final iff it is a τ -stable configuration, the computational devices on all its tiles are in a final state, and no more attachment or signal transitions are possible.

The following example shows a smart-TAM system at temperature 1, with a single tile type, that can assemble a linear supertile consisting of a row of three tiles. Without the local tile computational devices, such a tile system in the abstract tile assembly model, aTAM, could only generate an infinite linear supertile.

Example 2. Consider a smart-TAM system at temperature 1 that consists of a single smart tile, which is also its seed tile, $t_1 = (G_1, L_1, \Pi_1, C_1)$, see Figure 5.2 (1). The tile t_1 is defined such that $G_1 = \{(a, E, on), (a, W, on), (s, E, on), (s, W, off)\}$, the strength of glue a is 1, the strength of glue s is 0, $L_1 = \{T\}$, $\Pi_1 = \emptyset$, and the computational device C_1 is the GSM with set of states $\{q_0, q_R, q_C, q_L, q_F\}$, the initial state q_0 , the set of final states $\{q_R, q_L, q_F\}$, input alphabet $\{a, s\} \times D$, output alphabet $\{(a, E, off), (s, W, on), (a, W, off)\}$, and transition function defined as follows:

$$\begin{aligned} (q_0, (a, W)) &\rightarrow \{q_R\} \times \{(a, E, off)\} \\ (q_0, (a, E)) &\rightarrow \{q_C\} \times \{(s, W, on)\} \\ (q_0, (s, E)) &\rightarrow \{q_L\} \times \{(a, W, off)\} \\ (q_C, (s, E)) &\rightarrow \{q_L\} \times \{(a, W, off)\} \\ (q_C, (a, W)) &\rightarrow \{q_F\} \times \emptyset. \end{aligned}$$

The seed tile t_1 can attach to two more tiles of same tile type, one at its left and one at its right, to assemble a final configuration consisting of a row of three tiles. In the beginning, the computational devices in all individual smart tiles are in their start state, q_0 . Figure 5.2 (2) illustrates an attachment transition of one tile to the right of the seed tile. The seed tile changes its state from q_0 to q_C , indicating that it will become the center tile in the final configuration. The

newly attached tile changes its state from q_0 to q_R , indicating that it will become the rightmost tile in the final configuration. Figure 5.2 (3) shows the result of applying a signal transition to the supertile in Figure 5.2 (2), that is, of applying the pending actions to that supertile. Note that glue a on the east edge of the rightmost tile is deactivated, and thus no tile can attach to the east side of this supertile. Figure 5.2 (4) shows the result of an attachment transition to the left of the supertile in Figure 5.2 (3). The newly attached tile changes its state from q_0 to q_L , indicating that this tile will be leftmost tile in the final configuration. Figure 5.2 (5) shows the result of applying a signal transition to the supertile in Figure 5.2 (4), that is, of applying all pending actions. This supertile is the final configuration since no more attachment or signal transitions are possible, and the GSMs of all tiles are in a final state.

There are three main differences between smart-TAM and STAM. First, smart tiles in smart-TAM are endowed with computational devices which allow signal reuse as well as having additional control over the self-assembly due to their internal states. Second, smart-TAM allows both activation and deactivation of glues, while STAM does not allow activation of deactivated glues. Third, smart-TAM as defined in this paper only allows the attachment of one tile at a time in each attachment transition, while STAM allows the attachment of two supertiles to each other during one attachment transition.

Informally, if STAM were restricted to the attachment of one tile at a time, then STAM would be a particular case of smart-TAM, with a “look-up table” as the local tile computational device, and no activation of deactivated glues. Alternatively, if one were to generalize smart-TAM to 2smart-TAM, that is, a smart tile assembly model exactly like the one in this paper except that it would use 2HAM-style attachments instead of only one-tile-at-a-time attachments, then STAM would be strictly weaker than such a 2smart-TAM.

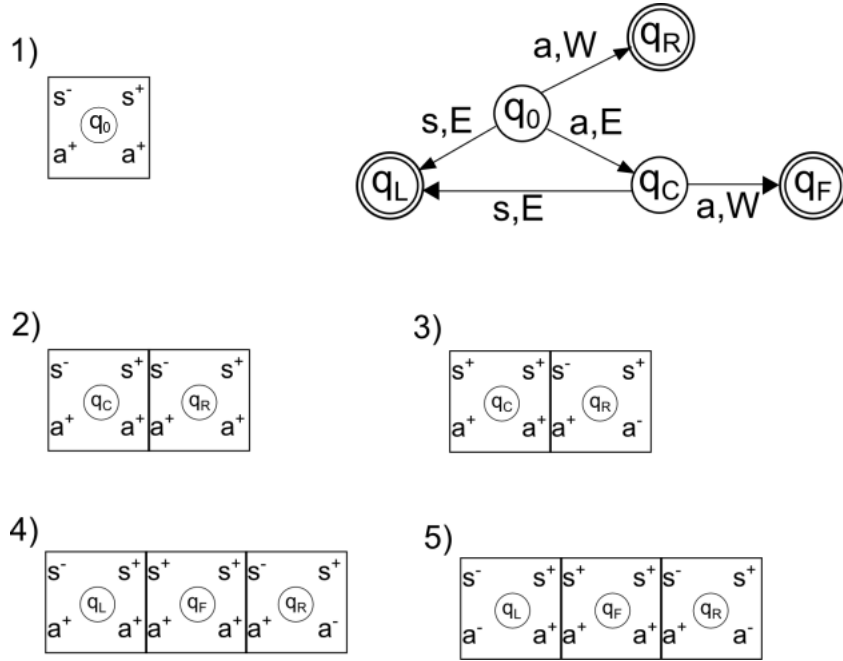


Figure 5.2: A smart-TAM system at temperature 1 with a single tile type that can assemble a final configuration consisting of a row of three tiles (Example 2). The state of the GSM of a smart tile is shown inside a circle at the center of the tile. The tile labels are not shown. The final states of the GSM are indicated by double circles around the states. (1) Left: The only smart tile type in the system; Right: The GSM of the smart tile. (2) The result of an attachment transition of one tile to the right of the seed tile. (3): The result of applying a signal transition to the supertile in (2), that is, of applying all the pending actions. (4): The result of an attachment transition of a tile to the left of the supertile in (3). (5): The final configuration obtained after applying a signal transition to the supertile in (4).

5.3 A Smart Tile Assembly System that Replicates L -Convex Shapes

In this section, we construct a smart tile assembly system that can replicate any L -convex shape, and indicate how it can be easily modified to perform arbitrary convex shape replications. We note that the replication is at a one-to-one scale, and that the constructed smart-TAM system (and implicitly its tile-complexity) is independent of the size and particular shape that it is replicating. The only requirement is the existence of an initial supertile of the desired shape, whose exterior edges contain minimal information about their being a north, east, south or west edge, or their respectively being edges of one of the four "corner tiles" (similar assumptions have been made in, e.g., [4]). We start by recalling some basic notions about 2D grids.

A *cell* on a 2D grid is an ordered pair (x, y) , where $x, y \in \mathbb{Z}$. The vectors $(0, 1)$, $(0, -1)$, $(-1, 0)$, and $(1, 0)$ are the unit vectors. Two cells $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$ are adjacent if $(x_1, y_1) = (x_2, y_2) + \vec{u}$ where \vec{u} is one of the unit vectors. A *shape* is, informally, a set of connected cells. Formally, a set of cells $C = \{(x, y) | x, y \in \mathbb{Z}\}$ is called a *shape* if for all pairs of distinct cells $c_1 \in C$ and $c_2 \in C$ there exists a sequence of unit vectors $p = (\vec{u}_1, \vec{u}_2, \dots, \vec{u}_n)$, $n \geq 1$, such that $c_2 = c_1 + \sum_{i=1}^n \vec{u}_i$ and, for all $1 \leq k \leq n$ we have $(c_1 + \sum_{i=1}^k \vec{u}_i) \in C$. A sequence of unit vectors $p = (\vec{u}_1, \vec{u}_2, \dots, \vec{u}_n)$ satisfying the conditions above is called a path from cell c_1 to cell c_2 , inside the shape C , and n is called the length of p .

Definition A shape C is called a convex shape if, for all pairs of cells $c_1, c_2 \in C$, there exists a path $p = (\vec{u}_1, \vec{u}_2, \dots, \vec{u}_n)$ inside C from the cell c_1 to the cell c_2 and, moreover, the cardinality of the set $\{\vec{u}_1, \vec{u}_2, \dots, \vec{u}_n\}$ is 2. In other words, p contains at most two distinct types of unit vectors.

A convex shape C is called L -convex if, for all pairs of cells $c_1, c_2 \in C$, there exists a path p inside of C with no more than one change of direction.

Definition A convex shape C is called L -convex if, for all pairs of cells $c_1, c_2 \in C$, there exists a path $p = (\vec{u}_1, \vec{u}_2, \dots, \vec{u}_n)$, $n \geq 1$, inside the shape C , from cell c_1 to c_2 , such that either the length of the path is $n \leq 2$ or there exists a $1 \leq k < n$ such that $\vec{u}_1 = \dots = \vec{u}_k$ and $\vec{u}_{k+1} = \dots = \vec{u}_n$.

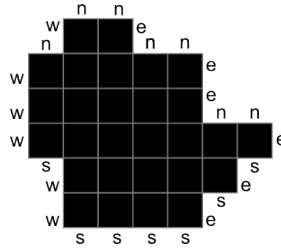


Figure 5.3: An example of an L -convex supertile. From every tile in the above supertile, all other tiles inside the supertile can be reached via a path inside the supertile that has maximum one change in direction. The edges labelled with n are north-free, the edges labelled with w are west-free, the ones labelled with e are east-free, and the ones labelled with s are south-free.

Given a supertile S , since the binding graph of S is connected, we have that $\text{dom}(S)$ is a shape. The set $\text{dom}(S)$ will be called the shape of S , and a supertile will be called L -convex if its shape is L -convex. Figure 5.3 shows an example of an L -convex supertile.

Conversely, given a shape $A \subseteq \mathbb{Z}^2$, a supertile S is said to have shape A iff $\text{dom}(S)$ either equals A , or can be obtained from A via a translation, rotation by a multiple of 90° , or reflection. When it is clear from the context, the mapping between a supertile domain and its shape will not be mentioned, and the same path definition will be used for both supertiles and shapes.

The north edge of a tile t from the supertile S is called north-free if there is no tile from S attached to the north edge of t . The east-free, west-free, and south-free edges of a tile t from the supertile S are similarly defined. Figure 5.3 illustrates the north-free, east-free, west-free, and south-free edges of an L -convex supertile.

A north-free edge of the tile t from the L -convex supertile S is called the *north-west edge* of S , if the tile t is the tile located on the west-most column of the north-most row of the supertile S . Formally, the north-free edge of a tile t from the L -convex supertile S , with coordinates (x, y) , is called the *north-west edge* of S if, for all tiles t' with coordinates (x', y') from S , we have that either $y' < y$ or $y' = y$ and $x' > x$. Moreover, the tile t that has the north-west edge of S is called the *north-west tile* of S . Similarly, the *east-north tile* is the north-most tile of the east-most column, the *south-east tile* of S is the east-most tile of its south-most row, the *west-south tile* is the south-most tile of west-most column. These tiles will be called the *corner tiles* of an L -convex supertile.

Lemma 5.3.1 *Every L -convex supertile S has at most four corner tiles: one north-west, one*

east-north, one south-east, and one west-south tile.

Lemma 5.3.2 *Let tile c_1 with coordinates (x_1, y_1) be the north-west tile of the L -convex supertile A , and tile c_2 with coordinates (x_2, y_2) be the east-north tile of A . Then, for all $x, y \in \mathbb{Z}$, if $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$ then the tiles with coordinates (x, y_2) and (x_1, y) are in A .*

Proof Because the supertile A is L -convex, there exists a path between c_2 and c_1 that lies inside A and that has only one change of direction. Since c_2 is the east-north tile and c_1 is the north-west tile of the supertile A , the directions on this path from c_2 to c_1 can only be north and west. Moreover, this path that starts from c_2 cannot start with the north direction, so it has to start in a west direction, move west until it reaches the x -coordinate x_2 , then turn north and move north until it reaches the y -coordinate y_2 . As a result, for all $x, y \in \mathbb{Z}$, if $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$ the tiles with coordinates (x, y_2) form the horizontal part of the path, while the tiles with coordinates (x_1, y) form the vertical part of the path between c_2 and c_1 , and all these lie within A because A is L -convex.

Corollary 5.3.3 *Statements analogous to Lemma 5.3.2 hold for the pairs of tiles east-north with south-east, south-east with west-south, and west-south with north-west, respectively.*

Our goal is to design a smart-tile assembly system at temperature $\tau = 2$ that can replicate any L -convex shape. That is, we construct a smart-tile assembly system that, given as input an L -convex supertile of the required shape, produces a second supertile that is identical to the original one, modulo a translation and a reflection. In contrast with [11], our construction makes no assumptions on the interior tiles of the input supertile. In particular, our construction does not require any specific glues, or signal transitions associated with the tiles from the interior of the input supertile. The only requirement is that the border tiles of the input supertile satisfy the following conditions: All the free edges (edges without attachment and belonging to a tile on the border of the supertile) on the north, east, south and west must have glues n , e , s , and w respectively. The exceptions are the north-west, east-north, south-east, and west-south edges of the supertile which must have the glues S_n , S_e , S_s , and S_w respectively. Figure 5.4 (left) shows an example of an L -convex supertile and the expected glues on its borders.

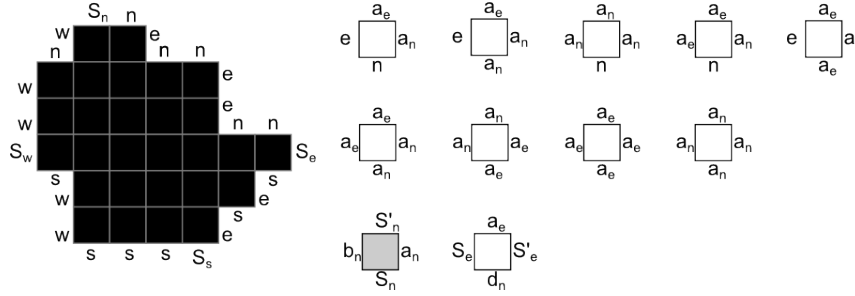


Figure 5.4: *Left:* An example of an L -convex supertile with the border glues needed for its replication by the smart-TAM system from Theorem 5.3.4. *Right:* The tile set $\Theta_{fill-ne}$, which fills the area at the north-east of the L -convex supertile, up to a rectangle. The tile marked in grey is the seed tile, s_{seed} . Glues S_n (on the north-east edge of the supertile), S_e (on the east-south edge of the supertile), S'_s (on the south-west edge of the supertile) and S'_w (on the west-north edge of the supertile) have strength equal to the temperature, $\tau = 2$, and the remaining glues have strength 1.

Theorem 5.3.4 *There exists a smart-TAM system $(\Theta, g, \tau, s_{seed})$ at temperature $\tau = 2$ with the following property: For any L -convex shape A , there exists an L -convex supertile A' with shape A such that, given A' as an input, the smart-TAM system self-assembles a second supertile with shape A .*

Proof Figure 5.5 shows the general idea of the replication process.

The self-assembly process starts from an L -convex supertile A' of the required shape A , with predefined glues on the border edges as described in the preamble of this theorem, and the seed tile s_{seed} (the tile marked in grey in Figure 5.4). The replication process starts with *Step 1*, filling in the areas on the four sides of the input supertile A' to form a rectangle that surrounds it. This is accomplished using the tile set $\Theta_{fill} = \Theta_{fill-ne} \cup \Theta_{fill-nw} \cup \Theta_{fill-se} \cup \Theta_{fill-sw}$. The borders of the rectangle that surrounds the input supertile encode information about the external border of the supertile itself. In *Step 2*, a different set of tiles, $\Theta_{transfer}$, is used to transfer the information from the borders of this rectangle to construct an identical rectangular hole underneath it. Following this, in *Step 3*, the set of tiles Θ_{border} fills in this rectangular hole, modulo a supertile-shaped hole in the middle. Finally, in *Step 4*, the set of tiles $\Theta_{replica}$ fills in the remaining supertile-shaped hole with a replica of the input supertile, and detaches this replica from the entire scaffold structure. Note that *Step 3* and *Step 4* are the only ones requiring the active use of the local computational devices on tiles, herein generalized sequential machines.

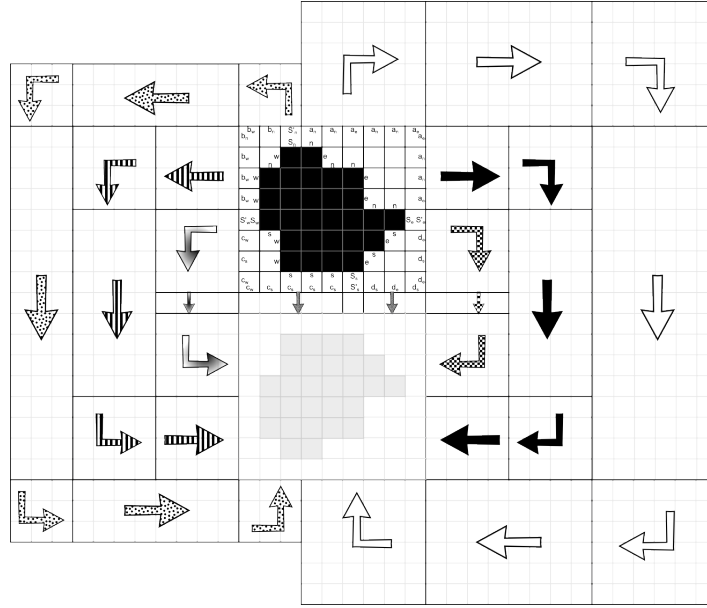


Figure 5.5: The general idea of the replication process of an L -convex supertile of a given shape, by a smart tile self-assembly system. The original shape is shown in black, and the constructed replica is in light grey. After tiles are used to fill in a rectangle that surrounds the original supertile (*Step 1*), information about the borders of the supertile (now encoded on the edges of the rectangle) is transported to form an identical rectangular hole underneath it (*Step 2*). The flow of this information is indicated by arrows. Afterwards, the rectangular hole is filled in, modulo a supertile-shaped hole (*Step 3*). Finally, the supertile-shaped hole is filled in with a replica of the input supertile, and the replica then detaches from the scaffold structure (*Step 4*).

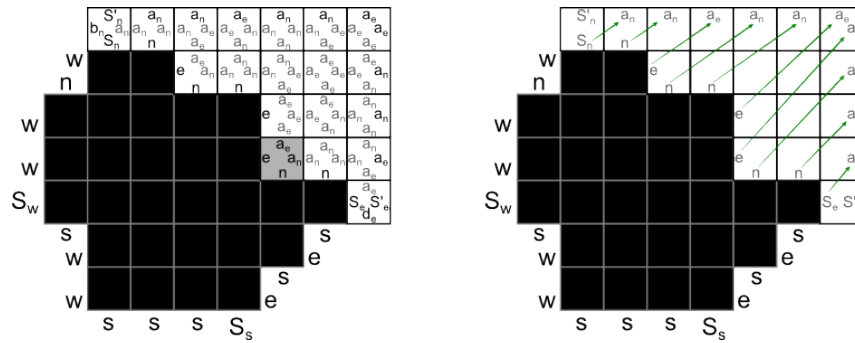


Figure 5.6: *Left*: The tile set $\Theta_{\text{fill-ne}}$ is used to fill the area at the north-east of the L -convex supertile (in black). The inside border of the assembled quarter-rectangle, read from top-left to bottom-right, starts with S'_n and ends with S'_e . The glues on the free edges of the white quarter-rectangle contain information about the “north-east” border of the supertile. *Right*: The arrows show the bijective mapping between the glues on the edges on the “north-east” border of the original supertile and the glues on the free edges of the tiles on the border of the quarter-rectangle.

b_w	b_n	S'_n	a_n	a_n	a_e	a_n	a_n	a_e
b_n		S_n	n					a_e
b_w	w			e	n			a_n
b_w	w					e		a_n
b_w	w					e	n	a_e
$S'_w S_w$								$S_e S'_e$
c_w	s	w					s	d_e
c_s		w				e	s	d_s
c_w		s	s	s	S_s			d_e
c_w	c_w	c_s	c_s	c_s	S'_s	d_s	d_e	d_s

Figure 5.7: The result of *Step 1*: Filling all four quarter-rectangles around the original supertile from Figure 5.4. The glues on the border of the resulting rectangle correspond to the glues on the free edges of the original supertile.

The details of the construction are described below.

Step 1: Figure 5.4 (right) shows the tiles in $\Theta_{fill-ne}$ that fill the north-east corner of the rectangle that will surround the input supertile. The first row consists of the tiles that attach to the border of the supertile, while the second and the third row are the tiles that fill the remaining area of the rectangle. All glues of the tiles in $\Theta_{fill-ne}$ have temperature 1, except glues of the north-west edge S_n and that of the east-north edge, S_e , which have temperature 2.

The role of the tiles in $\Theta_{fill-ne}$ is to transmit the information about the border of the supertile towards the edges of the rectangle. They transmit the information from their south edge to the east, and the information from their west edge to the north. For example, the first tile on the first row in Figure 5.4 (right) has glue n on its south edge and glue e on its west edge, which means it attaches to the supertile as illustrated in Figure 5.6 (left) (the grey tile). To transmit the information n coming from its south edge, the east edge of this tile has glue a_n . Similarly, to transmit information e coming from its west edge, the north edge of the grey tile has glue a_e . Figure 5.6 (left) shows the result of the attachment of the tile set $\Theta_{fill-ne}$ to the L -convex supertile from Figure 5.4. Figure 5.6 (right) shows the mapping between the glues on the border of the supertile and the glues on the edges of the tiles of the filled rectangle. The result of *Step 1* is shown in Figure 5.7.

Step 2: The arrows in Figure 5.5 show the general directions in which the self-assembly proceeds, that results in the transfer of the information from the borders of the rectangle that

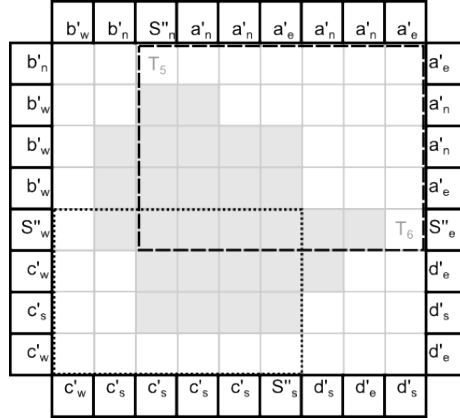


Figure 5.8: The rectangular hole that is obtained at the end of *Step 2*. The tiles that border the rectangular hole are outlined in thick lines: The inside glues of these tiles are identical in information content to the glues on the free edges of the tiles on the outside border of the rectangle that surrounded the input supertile. The thin lines inside the rectangular hole are a preview of the subsequent steps, which will first (*Step 3*) fill up the rectangular hole, modulo a supertile-shaped hole (light grey). This will be followed (*Step 4*) by filling the supertile-shaped hole with a replica. Tiles T_5 and T_6 will guide this process in the north-east quarter-rectangle. Note that the north-east quarter-rectangle (dashed lines) can overlap with, e.g., the south-west quarter-rectangle (dotted lines).

surrounds the supertile to the borders of a rectangular “hole”, underneath it. The tiles on the inside border of this rectangular hole encode information about the edges of the input supertile, and this hole will be the place where the replica of the input supertile will be assembled. The strength of all the glues on this inner border of the rectangular hole is 1. The construction of the tile set $\Theta_{transfer}$, that assembles squares and rectangles which transfer information about the input supertile’s border glues horizontally, vertically, or at 90° (Figure 5.5), is standard, see, e.g., [3] or [20], and will be omitted. Note that all the squares that transfer information at 90° (see Figure 5.5), that is, “turn corners”, can be built using a constant number of tiles, similar to [3]. The rectangles that transfer information vertically or horizontally (marked with horizontal or vertical thick arrows in Figure 5.5) have sizes determined by the adjacent squares and the border of the original supertile, and thus the tile set that constructs these rectangles does not need any prior information regarding their size. The result of this step is illustrated in Figure 5.8.

Step 3: The purpose of this step is to fill in the rectangular hole obtained in *Step 2*, until only a smaller hole remains in the middle, shaped exactly like the input supertile. The smart

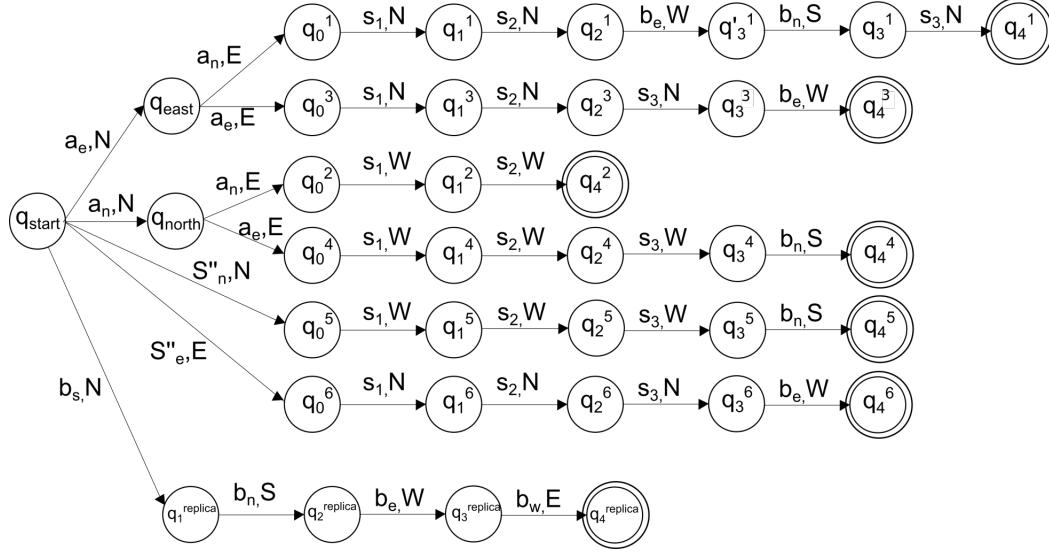


Figure 5.10: The local tile computational device (GSM). Only the component that deals with the north-east rectangle is shown. The complete input and outputs for each transition are listed in the text.

Initial transitions : $(q_{start}, (a_e, N)) \rightarrow \{q_{east}\} \times \emptyset$

$(q_{start}, (a_n, N)) \rightarrow \{q_{north}\} \times \emptyset$

$(q_{east}, (a_n, E)) \rightarrow \{q_0^1\} \times \emptyset$

$(q_{east}, (a_e, E)) \rightarrow \{q_0^3\} \times \emptyset$

$(q_{north}, (a_n, E)) \rightarrow \{q_0^2\} \times \emptyset$

$(q_{north}, (a_e, E)) \rightarrow \{q_0^4\} \times \emptyset$

$T_1 : (q_0^1, (s_1, N)) \rightarrow \{q_1^1\} \times \{(a'_e, W, \text{off}), (a'_n, S, \text{off}), (s_1, E, \text{on})\}$

$(q_1^1, (s_2, N)) \rightarrow \{q_2^1\} \times \{(f, W, \text{on}), (f, S, \text{on}), (s_2, E, \text{on})\}$

$(q_2^1, (b_e, W)) \rightarrow \{q_3'^1\} \times \emptyset$

$(q_3'^1, (b_n, S)) \rightarrow \{q_3^1\} \times \emptyset$

$(q_3^1, (s_3, N)) \rightarrow \{q_4^1\} \times \{(s_4, W, \text{on}), (s_4, S, \text{on}), (s_3, E, \text{on})\}$

$T_2 : (q_0^2, (s_1, W)) \rightarrow \{q_1^2\} \times \{(s_1, S, \text{on})\}$

$(q_1^2, (s_2, W)) \rightarrow \{q_4^2\} \times \{(s_2, S, \text{on})\}$

$$T_3 : (q_0^3, (s_1, N)) \rightarrow \{q_1^3\} \times \{(a'_e, W, \text{off}), (s_1, S, \text{on})\}$$

$$(q_1^3, (s_2, N)) \rightarrow \{q_2^3\} \times \{(f, W, \text{on}), (s_2, S, \text{on})\}$$

$$(q_2^3, (s_3, N)) \rightarrow \{q_3^3\} \times \emptyset$$

$$(q_3^3, (b_e, W)) \rightarrow \{q_4^3\} \times \{(s_4, W, \text{on}), (s_3, S, \text{on})\}$$

$$T_4 : (q_0^4, (s_1, W)) \rightarrow \{q_1^4\} \times \{(a'_n, S, \text{off}), (s_1, E, \text{on})\}$$

$$(q_1^4, (s_2, W)) \rightarrow \{q_2^4\} \times \{(f, S, \text{on}), (s_2, E, \text{on})\}$$

$$(q_2^4, (s_3, W)) \rightarrow \{q_3^4\} \times \emptyset$$

$$(q_3^4, (b_n, S)) \rightarrow \{q_4^4\} \times \{(s_4, S, \text{on}), (s_3, E, \text{on})\}$$

$$T_5 : (q_{\text{start}}, (S''_n, N)) \rightarrow \{q_0^5\} \times \{(s_1, E, \text{on})\}$$

$$(q_0^5, (s_1, W)) \rightarrow \{q_1^5\} \times \{(f, S, \text{on}), (s_2, E, \text{on})\}$$

$$(q_1^5, (s_2, W)) \rightarrow \{q_2^5\} \times \{(s_3, E, \text{on})\}$$

$$(q_2^5, (s_3, W)) \rightarrow \{q_3^5\} \times \{(s_3, E, \text{on})\}$$

$$(q_3^5, (b_n, S)) \rightarrow \{q_4^5\} \times \{(s_4, E, \text{on})\}$$

$$T_6 : (q_{\text{start}}, (S''_e, E)) \rightarrow \{q_0^6\} \times \emptyset$$

$$(q_0^6, (s_1, N)) \rightarrow \{q_1^6\} \times \{(f, W, \text{on}), (s_1, S, \text{on})\}$$

$$(q_1^6, (s_2, N)) \rightarrow \{q_2^6\} \times \{(s_2, S, \text{on})\}$$

$$(q_2^6, (s_3, N)) \rightarrow \{q_3^6\} \times \{(s_3, E, \text{on})\}$$

$$(q_3^6, (b_n, W)) \rightarrow \{q_4^6\} \times \{(s_4, E, \text{on})\}$$

$$\begin{aligned}
T_{\text{replica}} : (q_{\text{start}}, (f_s, N)) &\rightarrow \{q_{\text{start}}\} \times \{(f_1, S, \text{on})\} \\
(q_{\text{start}}, (b_s, N)) &\rightarrow \{q_1^{\text{replica}}\} \times \{(b_s, S, \text{on})\} \\
(q_1^{\text{replica}}, (b_n, S)) &\rightarrow \{q_2^{\text{replica}}\} \times \{(b_n, N, \text{on})\} \\
(q_2^{\text{replica}}, (b_e, W)) &\rightarrow \{q_3^{\text{replica}}\} \times \{(b_e, E, \text{on})\} \\
(q_3^{\text{replica}}, (b_w, E)) &\rightarrow \{q_4^{\text{replica}}\} \times \{(b_e, E, \text{on})\} \\
(q_4^{\text{replica}}, (s_4, E)) &\rightarrow \{q_4^{\text{replica}}\} \times \{(f, E, \text{off})\} \\
(q_4^{\text{replica}}, (s_4, W)) &\rightarrow \{q_4^{\text{replica}}\} \times \{(f, W, \text{off})\} \\
(q_4^{\text{replica}}, (s_4, N)) &\rightarrow \{q_4^{\text{replica}}\} \times \{(f, N, \text{off})\} \\
(q_4^{\text{replica}}, (s_4, S)) &\rightarrow \{q_4^{\text{replica}}\} \times \{(f, S, \text{off})\}
\end{aligned}$$

Note that the first two transitions of the GSM in tiles T_i , $1 \leq i \leq 4$ are applied during the two attachment transitions that attach the tile to the assembly and that, after this attachment stage, the GSM's in all tiles in the north-east quarter-rectangle are in a state q_0^j , $1 \leq j \leq 4$.

The role of the “signal” s_1 (a signal is a glue with strength 0) is to carve a supertile-shaped hole from the rectangle. It starts in tile T_5 (see Figure 5.8) and travels clockwise through the tiles right outside of the replica, deactivating the glues on the replica border tiles, and changing their GSM states to q_1^j , $1 \leq j \leq 6$. After arriving in T_6 , this signal continues travelling through the other three quarter-rectangles, deactivating the glues of the replica border tiles on its way, until it arrives back to T_5 . At this moment, the supertile-shaped middle of the rectangle detaches completely, leaving a supertile-shaped hole.

At this point, tile T_5 activates the “signal” s_2 (glue with strength 0). This signal then travels clockwise through the same path as s_1 , along the tiles right outside the supertile-shaped hole, making their inside edges “sticky”, and changing the states of their GSM's to q_2^j , $1 \leq j \leq 6$. This is in preparation for the next step (the assembly of the supertile-shaped replica). Making the inside edges of the tiles sticky is achieved by the use of a single new glue, f , which is activated by s_2 . At this moment, all computational devices in the tiles T_1, T_3, T_4 in the north-east quarter-rectangle are in state q_2^j , $j \in \{1, 2, 4\}$. Moreover all tiles T_2 in the north-east quarter rectangle are in state q_4^2 .

At the end of *Step 3*, we have a rectangle underneath that input supertile, which has a supertile-shaped hole with sticky edges in the middle.

The fact that the input supertile is L -convex is essential for *Step 3*. Due to the L -convexity of the shape, Lemma 5.3.2 guarantees that the attachment of the tiles T_1 through T_4 from $\Theta_{border-ne}$ does not interfere with the construction of the border of the supertile-shaped hole in the other quarter-rectangles. Indeed, the one problem that could have arisen in this step that, because the system is non-deterministic, the four quarter-rectangles can assemble independently and potentially overlap (see Figure 5.8) and this could interfere with signal transmission. However, we note that the inside “straight” borders of these quarter-rectangles coincide with the path between two of the “corner” tiles of the replica that was mentioned in Lemma 5.3.2, augmented with two end-tiles. Due to Lemma 5.3.2, this entire path lies on the inside of the replica. Together, these guarantee that any intersection between two quarter-rectangles lies inside the replica, and since both signals s_1 and s_2 travel right outside the replica, this implies that the paths of the signals never fall inside an overlap. Thus, any such overlap is unimportant, since it does not interfere with signal transmission and, moreover, any overlap tiles will anyway be all detached in the process of carving the supertile-shaped hole.

Step 4. At the start of this step, tile T_5 activates signal s_3 (which will later be used for starting the detachment of the constructed replica). In this last step, the one tile from the tile set $\Theta_{replica}$, see Figure 5.9, fills in the supertile-shaped hole as follows.

Tile $T_{replica}$ has glue f^+ and f_1^- as well as glues b_n, b_e, b_s, b_w , on all of its edges. Glue f has strength 1 and is used to attach each tile inside the replica to its neighbours in the replica. Glue f_1 has strength 1 and is used to make sure that the replica is τ -stable, by strengthening the power of the attachment of the corner tile to the replica (see transitions corresponding to $T_{replica}$). Glues b_n, b_e, b_s, b_w have strength 0 (are signals), and are used to guarantee that the replica is hole-free. Indeed, signal b_n is sent by each tile from the south edge of the replica to the north, and b_s is sent in the opposite direction. Similarly, signal b_e is sent from each tile on the west edge of the replica to the east edge, and b_w is sent in the opposite direction. The fact that these signals have all reached the opposite edge means that the supertile-shaped hole is completely filled in. This fact is recorded by the GSM’s of the tiles situated immediately outside the replica, on its outside border, changing their state from q_2^j to q_3^j , $1 \leq j \leq 6$ (via

state $q_3'^1$ in the case of T_1). In addition, when each tile in the replica has passed all four signals, b_n, b_e, b_s, b_w , the state of its GSM changes to $q_4^{replica}$, a final state.

When signal s_3 , after having started in tile T_5 , comes back to T_5 , after having travelled clockwise, right outside the border of the replica, if the GSM of T_5 is in state q_3^5 (indicating that the supertile-shaped hole has been filled with the replica) then tile T_5 initiates signal s_4 which travels along the outside border of the replica and deactivates the outside glues of the tiles on the border of the replica. As result of this second part of *Step 4* the replica, whose tiles all have their GSM's in a final state, detaches from the scaffold structure.

Note that the replica of the input supertile is τ -stable. Indeed, all the tiles in the replica are from the tile set $\Theta_{replica}$ and have at least two common edges with the tiles from the same tile set, with maximum four exceptions. These exceptions are the tiles that are attached to the four special “border” tiles (representatives of the north-west, north-east, south-east and south-west edges), as these tiles might have only one common edge with tiles from the replica. However, the attachments between these special tiles and the replica have been strengthened by the activation of the glue f_1 , and now suffice to keep such special tiles attached to the replica.

The smart-TAM system in Theorem 5.3.4 can be used, with slight modifications, to replicate any convex shape (not necessarily L -convex). Indeed, the only part in the construction that relied on the input supertile being L -convex was *Step 3*, that fills in the rectangular hole in which the replication will take place. Here, the problem was the potential overlap between the north-east, south-east, south-west and north-west quarter-rectangles (Figure 5.8) could potentially interfere with the signal transmission. In our construction, this problem was avoided by the assumption of L -convexity and Lemma 5.3.2 and Corollary 5.3.3. However, this situation can also be handled in a different way, by using (additional) different signals in each of the quarter rectangles. We did not use this solution, and preferred adding the L -convex assumption instead, because using more signals would have lead to the GSM's having more states and being more complex.

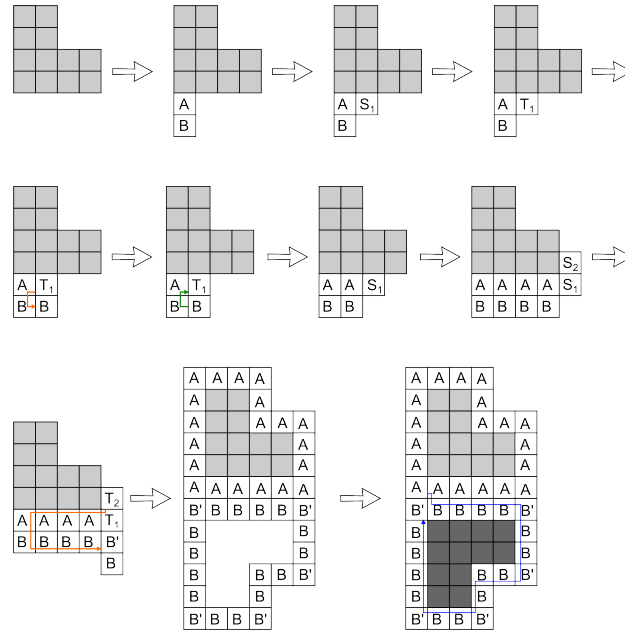


Figure 5.11: High-level design of another smart tile assembly system that replicates arbitrary shapes, and which uses a method similar to the smart sand robotic system replication of shapes [8].

5.4 Discussion and Future Work

Note that the tile complexity (number of tiles) and the space complexity (the “area” of the computation) of the construction of the smart-TAM replicating systems in Theorem 5.3.4 can be further reduced if we use a more complex tile computational device. The main idea is illustrated in Figure 5.11. Basically, in order to replicate the original supertile, the smart-TAM system should be able to read the edges and send signals accordingly. During each phase of the replication, the system attaches a new tile (labelled with “A”) to the border of the original supertile that reads the glue on one of the free edges of a tile on the border of the original supertile, and sends a signal to the position where this edge will be replicated. Unlike the construction in this paper, this signal has to travel back to the next free-edge on the original supertile, and be reused to repeat the process. The complexity of the GSM will increase accordingly, as it has to keep track of the position of the last tile from where the signal was sent. When the border structure (labelled with “B”) is complete, the interior is filled in, forming a replica of the original shape. The tiles that fill the inside of the supertile-shaped hole can be similar to the tile T_{replica} . This smart-TAM system for the replication of arbitrary shapes has

lower tile and space complexity than the smart-TAM system constructed in Theorem 5.3.4, but this comes at the expense of an increased complexity of the computational devices on the tiles.

Acknowledgements: This research was supported by a Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant to L.K.

Bibliography

- [1] Z. Abel, N. Benbernou, M. Damian, E. D. Demaine, M. L. Demaine, R. Y. Flatland, S. D. Kominers, and R. T. Schweller. Shape replication through self-assembly and rnaase enzymes. In M. Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1045–1064. SIAM, 2010.
- [2] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro. An autonomous molecular computer for logical control of gene expression. *Nature*, 429(6990):423–429, 2004.
- [3] Y. Brun. Improving efficiency of 3-sat-solving tile systems. In Y. Sakakibara and Y. Mi, editors, *DNA Computing and Molecular Programming, DNA 16*, volume 6518 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2010.
- [4] C. Chalk, E. D. Demaine, M. L. Demaine, E. Martinez, R. Schweller, L. Vega, and T. Wylie. Universal shape replicators via self-assembly with attractive and repulsive forces. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA '17*, pages 225–238, Philadelphia, PA, USA, 2017.
- [5] Q. Cheng, G. Aggarwal, M. H. Goldwasser, M.-Y. Kao, R. T. Schweller, and P. M. de Espinosa. Complexities for generalized models of self-assembly. *SIAM Journal on Computing*, 34:1493–1515, 2005.
- [6] E. D. Demaine, M. L. Demaine, S. P. Fekete, M. Ishaque, E. Rafalin, R. T. Schweller, and D. L. Souvaine. Staged self-assembly: nanomanufacture of arbitrary shapes with $o(1)$ glues. *Natural Computing*, 7(3):347–370, 2008.

- [7] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Universal shape formation for programmable matter. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016*, pages 289–299, 2016.
- [8] K. Gilpin and D. Rus. A distributed algorithm for 2d shape duplication with smart pebble robots. In *Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2012*, pages 3285–3292. IEEE, 2012.
- [9] J. Hendricks, M. J. Patitz, and T. A. Rogers. Replication of arbitrary hole-free shapes via self-assembly with signal-passing tiles. In C. S. Calude and M. J. Dinneen, editors, *UCNC 2015, Proceedings*, volume 9252 of *Lecture Notes in Computer Science*, pages 202–214. Springer, 2015.
- [10] N. Jonoska and D. Karpenko. Active tile self-assembly, part 1: universality at temperature 1. *International Journal of Foundations of Computer Science*, 25(2):141–164, 2014.
- [11] A. Keenan, R. T. Schweller, and X. Zhong. Exponential replication of patterns in the signal tile assembly model. *Natural Computing*, 14(2):265–278, 2015.
- [12] K. Komiya, K. Sakamoto, H. Gouzu, S. Yokoyama, M. Arita, A. Nishikawa, and M. Hagiya. Successive state transitions with I/O interface by molecules. In A. Condon and G. Rozenberg, editors, *6th International Workshop on DNA-Based Computers, DNA 2000*, volume 2054 of *Lecture Notes in Computer Science*, pages 17–26. Springer, 2000.
- [13] A. Naz, B. Piranda, J. Bourgeois, and S. C. Goldstein. A distributed self-reconfiguration algorithm for cylindrical lattice-based modular robots. In A. Pellegrini, A. Gkoulalas-Divanis, P. di Sanzo, and D. R. Avresky, editors, *15th IEEE International Symposium on Network Computing and Applications, NCA 2016*, pages 254–263. IEEE Computer Society, 2016.
- [14] H. Oh, A. R. Shirazi, C. Sun, and Y. Jin. Bio-inspired self-organising multi-robot pattern formation: A review. *Robotics and Autonomous Systems*, 91:83–100, 2017.

- [15] J. E. Padilla, W. Liu, and N. C. Seeman. Hierarchical self assembly of patterns from the robinson tilings: DNA tile design in an enhanced tile assembly model. *Natural Computing*, 11(2):323–338, 2012.
- [16] J. E. Padilla, M. J. Patitz, R. Pena, R. T. Schweller, N. C. Seeman, R. Sheline, S. M. Summers, and X. Zhong. Asynchronous signal passing for tile self-assembly: fuel efficient computation and efficient assembly of shapes. In *Proceedings UCNC 2013*, pages 174–185, 2013.
- [17] M. J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014.
- [18] L. Qian and E. Winfree. Scaling up digital circuit computation with dna strand displacement cascades. *Science*, 332(6034):1196–1201, 2011.
- [19] M. Rubenstein, A. Cornejo, and R. Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.
- [20] S. Seki. Combinatorial optimization in pattern assembly - (extended abstract). In G. Mauri, A. Dennunzio, L. Manzoni, and A. E. Porreca, editors, *UCNC 2013, Proceedings*, volume 7956 of *Lecture Notes in Computer Science*, pages 220–231. Springer, 2013.
- [21] J. Werfel. Building blocks for multi-robot construction. In R. Alami, R. Chatila, and H. Asama, editors, *Distributed Autonomous Robotic Systems 6*, pages 285–294, Tokyo, 2007. Springer Japan.
- [22] J. Werfel. Collective construction with robot swarms. In R. Doursat, H. Sayama, and O. Michel, editors, *Morphogenetic Engineering, Toward Programmable Complex Systems*, pages 115–140. Springer, 2012.
- [23] E. Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, 1998.

Chapter 6

Conclusion

The goal of this research was to enrich our understanding of self-assembly of tiles by defining and studying theoretical models of self-assembly, and making a connection between research on tile-based self-assembly systems and other research areas such as robotics and the formal language theory. Compared to formal language theory and robotic systems, tile-based assembly systems are a relatively new research area. To be able to use theoretical results in other fields, a mapping between models is needed. We started this research project with an attempt to make a connection between patterns in tile-based self-assembly systems and two-dimensional languages. The definition of SA-hypergraph automata in Chapter 3 is a step in this direction. However, we believe many unanswered questions are left, and a deeper study of hypergraph automata can result in more interesting connections between self-assembly systems and picture languages. Since tiles in DNA-based self-assembly systems do not always have different properties or colors, one of the interesting topics to study is the relationship between the hypergraph automata and local picture languages. Also, many self-assembly systems do not have borders or have a partial border on one side. In Chapter 3, we only used framed picture languages, and an investigation into the picture languages that do not have frame or have partial frames, and their connection to self-assembly systems, would be a potential future area of research.

In the Chapter 4, it was shown that a small change in the definition of the tile assembly model can result in significantly different results, such as improving the computational power. Considering that we proved that glue deactivation is a powerful tool, it would be interesting to have a comparison between self-assembly systems using glue-deactivation-only signals and

self-assembly systems that use glue-activation-only signals. Both models are capable of simulating a Turing machine at temperature one. However, it is not known if there is any shape that can only be constructed using glue-activation-only models, and not by using glue-activation-only models. Moreover, it would be interesting to investigate the detachable tile assembly model from the point of view of Kolmogorov complexity.

Since there previously was no common framework for self-assembly models, no standard way to compare different self-assembly models was available. Our research in Chapter 5 aimed to define the smart tile assembly model, that can be used later as a general framework for self-assembly systems. Indeed, as a computational device on the smart tiles can range from being absent, to a simple counter, all the way to a Turing machine, it would be possible to frame and compare the existing self-assembly systems in terms of smart tile assembly models.

Lastly, we used smart tile systems to replicate given shapes. Although there have been other researchers who achieved shape replication using DNA-based tile assembly systems, none of the existing systems perform the replication task without moving the original shape from its place. This is important because replication of three-dimensional objects is not possible by building a frame around them, since the frame would stop the original shape from leaving its place. The fact that in our proposed system the original shape does not need to leave the frame that is built around it puts our proposed system in a unique position to be scalable to a three-dimensional version self-assembly. Indeed another direction of future work would be to extend our replication system to 3D

We hope the models that were proposed in this research can be used to apply research results from language theory to tile-based self-assembly systems and from tile-based self-assembly systems to robotics.

Appendix A

Examples of SA-Hypergraph Automata

In this section, we provide three example SA-hypergraph automata and illustrate their relation to self-assembly systems. Our findings, presented in Section 3.4, do not build upon this section. In all examples, every node in the underlying graph has a distinct colour which, for simplicity, is the same as the identifier of the node.

The following examples show an SA-hypergraph automaton to accept the pictures in Figure A.1 part *a*). This example shows that SA-hypergraph automata can accept a picture language with a simple description. The SA-hypergraph automaton in this example has 8 nodes and 3 hyperedges; the equivalent tile system needs 8 tile types.

Example The SA-hypergraph automaton for the example in Figure A.1 is defined as follows. The SA-hypergraph automaton is $A = (N, E, f, d, G, E_0)$, where

- $N = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$,
- $E = \{e_1, e_2, e_3\}$,
- function f is defined such that

$$f(e_1) = \{x_1, x_2, x_3, x_4\},$$

$$f(e_2) = \{x_3, x_4, x_5, x_6\},$$

$$f(e_3) = \{x_3, x_4, x_7, x_8\}$$

- function d is defined such that

$$d(e_1) = \{x_1, x_2\} \rightarrow \{x_3, x_4\},$$

$$d(e_2) = \{x_3, x_4\} \rightarrow \{\},$$

$$d(e_3) = \{x_3, x_4\} \rightarrow \{\}$$

- underlying graph is shown in Figure A.1 part b.
- $E_0 = \{e_1\}$

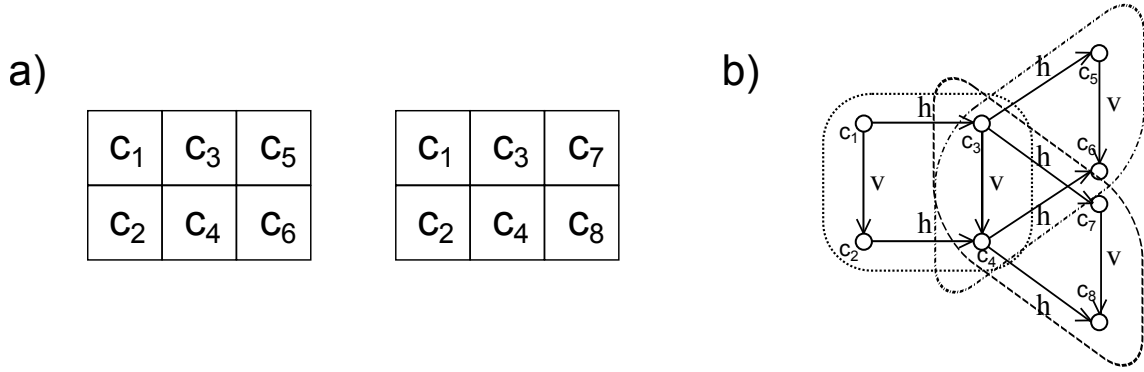


Figure A.1: Part a) shows an example of language of coloured self-assembled patterns. Part b) depicts the underlying graph of the SA-hypergraph automaton that constructs the same pattern.

Example A shows a simple picture language containing two 2D-words. The SA-hypergraph automaton uses two overlapping hyperedges with different active inputs and outputs. Therefore, the number of nodes in this SA-hypergraph automaton will be less than the number tiles in a tile assembly system which recognizes the same language. The SA-hypergraph automaton in this example has 4 nodes and 3 hyperedges. An equivalent tile assembly system needs at least 6 tile types.

Example The SA-hypergraph automaton for the example in Figure A.2 is defined as follows. The SA-hypergraph automaton is $A = (N, E, f, d, G, E_0)$, where

- $N = \{x_1, x_2, x_3, x_4\},$

- $E = \{e_1, e_2, e_3\}$,
- function f is defined such that

$$f(e_1) = \{x_1, x_2\},$$

$$f(e_2) = \{x_1, x_2\},$$

$$f(e_3) = \{x_1, x_2, x_3, x_4\}$$

- function d is defined such that

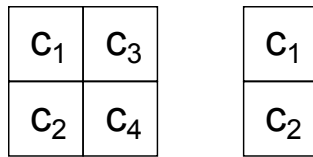
$$d(e_1) = \{x_1\} \rightarrow \{\},$$

$$d(e_2) = \{x_1\} \rightarrow \{x_1, x_2\},$$

$$d(e_3) = \{x_1, x_2\} \rightarrow \{\}$$

- underlying graph is shown in Figure A.2 part b).
- $E_0 = \{e_1, e_2\}$

a)



b)

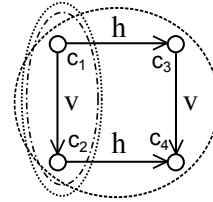


Figure A.2: Part a) shows an example of language of coloured self-assembled patterns. Part b) depicts the underlying graph of the SA-hypergraph automaton that constructs the same pattern.

Example A shows a language with an infinite number of one dimensional pictures. The SA-hypergraph automaton uses three hyperedges to build the chain, moreover, one more hyperedge is used to make the final configurations. Therefore, the number of nodes in this SA-hypergraph automaton will be less than the number tiles in a tile assembly system which recognizes the same language. The SA-hypergraph automaton in this example has 3 nodes and 4 hyperedges.

Whereas an equivalent tile assembly system needs at least 5 tile types (one tile type to start, 3 tile type to build the chain, and one tile type to stop).

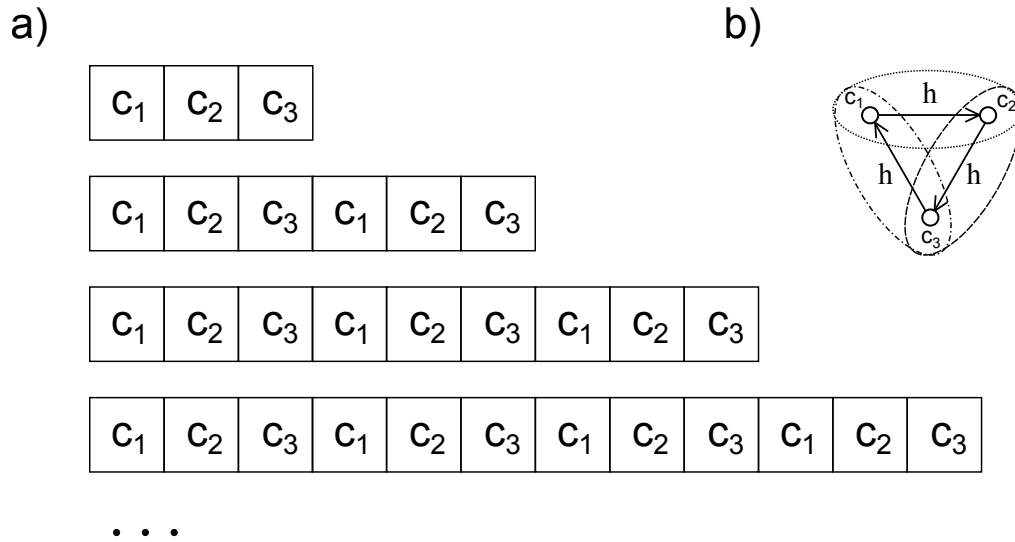


Figure A.3: Part a) shows an example of language of coloured self-assembled patterns. Part b) depicts the underlying graph of the SA-hypergraph automaton that constructs the same pattern.

Example The SA-hypergraph automaton for the example in Figure A.3 is defined as follows.

The SA-hypergraph automaton is $A = (N, E, f, d, G, E_0)$, where

- $N = \{x_1, x_2, x_3\}$,
- $E = \{e_1, e_2, e_3, e_4\}$,
- function f is defined such that

$$f(e_1) = \{x_1, x_2\},$$

$$f(e_2) = \{x_2, x_3\},$$

$$f(e_3) = \{x_3, x_1\}$$

$$f(e_4) = \{x_2, x_3\},$$

- function d is defined such that

$$d(e_1) = \{x_1\} \rightarrow \{x_2\},$$

$$d(e_2) = \{x_2\} \rightarrow \{x_3\},$$

$$d(e_3) = \{x_3\} \rightarrow \{x_1\}$$

$$d(e_4) = \{x_2\} \rightarrow \{\},$$

- underlying graph is shown in Figure A.3 part *b*).
- $E_0 = \{e_1\}$

Curriculum Vitae

Name: Amirhossein Simjour

Education and Degrees: University of Tehran
Tehran, Iran
2000 - 2005 B.Sc.

University of Tehran
Tehran, Iran
2010 - 2005 M.Sc.

University of Western Ontario
London, ON
2010 - 2017 Ph.D.

Related Work Experience: Teaching Assistant
The University of Western Ontario
2010 - 2014

Related publications:

1. Lila Kari, Steffen Kopecki, Amirhossein Simjour: Hypergraph Automata: a Theoretical Model for Patterned Self-assembly. Int. J. Found. Comput. Sci. 25(4): 419-440 (2014)
2. Lila Kari, Amirhossein Simjour: Smart Tile Self-Assembly and Replication. Fundam. Inform. 154(1-4): 239-260 (2017)

Publications under review:

1. Lila Kari, Amirhossein Simjour: Simplifying the Role of Signals in Tile Self-assembly.
(Submitted)