Self-Assembly on Various Surfaces

Shahrzad Heydarshahi

UNIVERSITÉ D'ORLÉANS

ÉCOLE DOCTORALE MATHÉMATIQUES, INFORMATIQUE, PHYSIQUE THÉORIQUE ET INGÉNIERIE DES SYSTÈMES

LABORATOIRE D'INFORMATIQUE FONDAMENTALE D'ORLÉANS

THÈSE présentée par :

Shahrzad HEYDARSHAHI

Soutenue le : 20 décembre 2023

pour obtenir le grade de : Docteur de l'Université d'Orléans

Discipline/Spécialité : Informatique

Self-Assembly on Various Surfaces

Auto-assemblage sur surfaces diverses

THÈSE dirigée par :

M. BECKER Florent	Maître de conférences, Université d'Orléans
M. DURAND-LOSE Jérôme	Professeur des Universités, Université d'Orléans

RAPPORTEURS :

M. FERNIQUE Thomas	Chargé de Recherche HDR, CNRS/Université Sorbonne Paris Nord
M. PATITZ Matthew John	Associate Professor, University of Arkansas

JURY :

M. BECKER Florent	Maître de conférences, Université d'Orléans, Encadrant de thèse
M. DURAND-LOSE Jérôme	Professeur des Universités, Université d'Orléans, Directeur de thèse
M. FERNIQUE Thomas	Chargé de Recherche HDR, CNRS/Université Sorbonne Paris Nord, Rapporteur
M. PATITZ Matthew John	Associate Professor, University of Arkansas, Rapporteur
M. REGNAULT Damien	Maître de conférences HDR, Université d'Évry Paris-Saclay, Examinateur
M. SENÉ Sylvain	Professeur des Universités, Université d'Aix-Marseille, **Président du jury**

Acknowledgements

I would like to express my deepest gratitude to my PhD supervisors, Jérôme and Florent, for their invaluable guidance and support throughout my PhD journey. Jérôme, your constant presence and the wisdom and kindness you shared have been a source of great learning and inspiration for me. Florent, thank you for being more a friend to me than an advisor. Your unwavering support over the years has meant so much to me. The best part of pursuing my PhD in Orléans was having the privilege of being supervised by both of you. Thank you.

I also extend my heartfelt thanks to my PhD referees, Matthew J. Patitz and Thomas Fernique, for their thorough reading and insightful comments on my manuscript, as well as their participation in my defense.

Further gratitude goes to Damien Regnault and Sylvain Sené for serving as examiners during my PhD defense.

I am deeply grateful to my former professors who inspired and encouraged me to pursue a PhD: in particular, Marc Chardin, Morteza Mohammad-Noori, Aline Parreau, Bernard Teissier, and Siamak Yassemi. I learned a lot from you, not only in science but also in morality and ethics. I was fortunate to have you as my guides during this journey.

I thank all my colleagues at LIFO, the Computer Science Department of COST, and the Computer Science Department of IUT. Special thanks to Anaïs and Benjamin for their unwavering support, as well as Ali, Anthony, Ioan, Jean-Michel, Jacques, Laure, Marta, Mathieu, Mirian, Noël, Pierre, Sébastien, Sophie, Sylvie, Wadoud and other colleagues. I also appreciate the technical and administrative staff of LIFO and the Computer Science Department for their assistance: Brigitte, Florence, Isabelle, Pascal, and Rali.

My thanks extend to my fellow PhD students, ATER, and postdocs at LIFO for the memorable times we shared, both in and out of LIFO, particular: Antoine, Aurélien, Benjamin, Florian, Giacomo, Hai-Yen, Imane, Kevin, Nedra , Sajad, Sofiane, Tenji, Thibaut. I would like to thank Darine for the good times that we had together. A special thanks to Maël and Samuel; our friendship has been one of the most rewarding aspects of my time at LIFO. I am also grateful to my friends and colleagues outside of Orléans, whom I met at conferences and other events, making these experiences enjoyable: Abeer, Ahmed, Cai and Carole.

I want to thank Katayoun Niloufari, my first French language teacher, who, when I was a teenager, sparked my interest in this language. I also extend my gratitude to the Embassy of France in Tehran and all its staff for supporting my first scientific visit to France at the end of my master's program in Tehran. In particular, I want to thank Mohammad Majlessi, who was very kind and helpful with my project of coming to France to continue my studies. I am grateful to the Fondation Sciences Mathématiques de Paris for their support and for preparing everything for starting my stay in France.

I want to give special thanks to my uncle, an incredible man whose presence in France gave me the chance to spend more time with him. Not only did I learn from his vast knowledge, but more importantly, I also enjoyed the simple pleasure of his company. Amoo Mohammad, I really appreciate what you did for me during all these years: thank you for everything.

I further thank the greatest gift in my life: my family. My mum, Minoo, that taught me patience and generosity for family love. My dad, Reza, the best father anyone could wish for and kindest person in the world for giving me always courage and confidence. My brother Amin, who is always ready to help and bring happiness to me; and Behnam, my caring brother, my friend and my soulmate that always makes me feel that someone

is beside me. I love you all so much. I also thank Bababozorg and Mamanbozorg, Didier, Gerda, Jurgen, Khale Mali, Nelly, and other family members for their love and support over these years. I have a special thanks to Mamanjoon, whose influence from living with her for years has shaped who I am. Your logic, intelligence, sense of humor, and poetry, along with your kindness and self-confidence, have always inspired me. You are always in my mind. Thank you for teaching me to be strong. I wish I could be like you in life.

Finally, I thank my friends: Aida, Aryana, Anahita, Azadeh, Barbara, Bertille, Elina, Elmira, Farbod, Farhang, Farshid, Fargol, Fatemeh, Hossein, Hiwa, Julien, Kaan, Laya, Leila, Mahdieh, Mahsa, Mahya, Maryam, Mina, Mitra, Mohammad, Mohammad Javad, Mona, Nairuhi, Narges, Niloufar, Parisa, Payam, Samaneh, Sanaz, Sara, Sahar, Shima, and Zohreh, and all my other friends who have helped and brought happiness to my life during these years. I also would like to express my special gratitude to Christelle, Nora, Karine, Solène and Tchakamé for all their support.

Writing an acknowledgment section is difficult, not only because it's easy to forget someone important but also because so many wonderful people have helped me along the way. It is impossible to mention all of them here, just as it is impossible to adequately describe the boundless love and support of my husband in mere words. All I can say is: my love, thank you for everything.

Auto-assemblage sur surfaces diverses

Résumé : Nous étudions l'auto-assemblage par tuiles sur divers types de surfaces. L'auto-assemblage par tuiles est un processus dynamique basé sur des *tuiles* carrées ayant quatre côtés distingués, qui peuvent s'attacher deux à deux sous certaines conditions. Le processus commence avec une configuration initiale appelée *graine*, et s'arrête si aucune autre tuile ne peut être ajoutée. Ce modèle peut être implémenté en pratique via les nanotechnologies de l'ADN, et il est étudié depuis la fin des années 1990. Il a été démontré que ce modèle permet de réaliser des calculs arbitraires : il s'agit d'un modèle de calcul universel. Nous étudions l'auto-assemblage par tuiles d'un point de vue théorique. La plupart des travaux sur ce sujet portent sur l'auto-assemblage par tuiles sur le plan Euclidien 2D. Nous nous intéressons à son comportement sur des surfaces plus complexes. La question centrale que nous posons est si l'on peut concevoir un système d'auto-assemblage par tuiles permettant de détecter le type de surface sur laquelle il est effectué ? Nous nous intéressons d'abord au cas de quatre types de *surfaces plates* : le tore plat, le cylindre vertical plat, le cylindre horizontal plat, et le plan Euclidien. Nous présentons un système d'auto-assemblage par tuiles dans le modèle classique *abstract Tile Assembly Model* (aTAM). Notre système peut être appliqué sur toute surface appartenant à l'un des quatre types de surfaces, et présente des particularités uniques en fonction du type de surface. Plus précisément, certaines tuiles apparaissent uniquement si l'assemblage a lieu sur l'un de ces types de surfaces. Nous abordons également des surfaces 3D plus complexes : celles d'objets 3D appelés *polycubes*. Les polycubes sont constitués de cubes unitaires collés les uns aux autres via leurs faces. La propriété la plus fondamentale d'une surfaces est sans doute son *genre* ; pour simplifier, il s'agit du nombre de trous qui perçent la surface. Notre but est de créer un système d'auto-assemblage par tuiles capable de détecter le genre du polycube sous-jacent. Afin d'effectuer de l'auto-assemblage par tuiles sur la surface d'un polycube, nous définissons tout d'abord un nouveau modèle adapté, appelé *Surface Flexible Tile Assembly Model* (SFTAM). Ce modèle étend le modèle aTAM et est inspiré par un autre modèle, le *Flexible Tile Assembly Model* (FTAM). Nous conçevons un système d'auto-assemblage par tuiles qui détecte le genre d'un type particulier de polycubes. Ces polycubes sont appelés *cuboïdes d'ordre 1*, et ils peuvent avoir genre 0 ou genre 1. Notre système est tel que, dans tout assemblage terminal, si le genre est 1, une tuile d'un ensemble particulier Y de types de tuiles apparaît nécessairement. Si le genre est 0, les tuiles dont le type est dans Y n'apparaissent dans aucun assemblage productible.

Mots clés : Auto-assemblage par tuiles, Calcul ADN, Surfaces géométriques, Genre, Polycubes

Résumé vulgarisé pour le grand public : Nous étudions l'auto-assemblage par tuiles sur divers types de surfaces. Il sagit d'un processus dynamique basé sur des *tuiles* carrées ayant quatre côtés distingués, qui peuvent se lier sous certaines conditions. Le processus commence par une configuration initiale appelée *graine*, et s'arrête si aucune autre tuile ne peut être ajoutée. Ce modèle peut être implémenté en pratique via les nanotechnologies de l'ADN, et il est l'objet de recherches depuis la fin des années 1990. Il a été démontré que c'est un modèle de calcul universel. La plupart des travaux du domaine portent sur l'auto-assemblage par tuiles sur le plan Euclidien. Nous nous intéressons à son comportement

sur des surfaces plus complexes. La question centrale que nous posons est si l'on peut concevoir un système d'auto-assemblage par tuiles permettant de détecter le type de surface sur laquelle il est effectué ? Nous répondons à cette question pour deux types de surfaces : les *surfaces plates* et certains *polycubes*.

Self-assembly on various surfaces

Abstract : We investigate tile self-assembly on various types of surfaces. Tile self-assembly is a dynamic process based on square *tiles* that have four distinguished sides that can attach to each other under certain conditions. The process starts with a specific predetermined configuration called a *seed*, and stops if no further tile can be attached. This model can be implemented in practice using DNA nanotechnology and has been investigated since the late 1990s. It has been shown theoretically to allow performing arbitrary computations : it is a universal computation model. We study tile self-assembly from the theoretical side. Most works in the area deal with the Euclidean 2D plane. We are interested in studying its behaviour on more complex surfaces. The central question that we ask, is, can we design a tile self-assembly system that can detect the type of surface that it is performed on? We first address this question for the case of four types of *flat surfaces*: the flat torus, the flat vertical cylinder, the flat horizontal cylinder, and the Euclidean plane. We design a tile self-assembly system in the classic *abstract Tile Assembly Model* (aTAM) that can be performed on any surface from one of these four types, and that exhibits specific features for each type. More precisely, certain tiles will uniquely appear if the assembly is taking place on one of these types of surfaces. We are also interested in more complex 3D surfaces that form the surface of objects called *polycubes*. Polycubes are made from unit cubes that are glued to each other by their faces. Perhaps the most fundamental property of a surface is its *genus*; informally, it is the number of holes piercing the surface. Our goal is to design a tile self-assembly system that can detect the genus of the underlying polycube surface. In order to perform tile self-assembly on polycubes, we first define a new suitable model, that we call the *Surface Flexible Tile Assembly Model* (SFTAM). This model extends the aTAM and is inspired by another existing model, the *Flexible Tile Assembly Model* (FTAM). We design a tile self-assembly system that detects the genus of a special type of polycubes. These polycubes are called *order-1 cuboids*, and they can have genus 0 or genus 1. In any terminal assembly of our system, if the genus is 1, a tile from a special subset Y of tile types must appear. When the genus is 0, however, tiles of Y never appear in any producible assembly.

Keywords : Tile self-assembly, DNA computing, Geometric surfaces, Genus, Polycubes

Abstract for general audience: We investigate tile self-assembly on various types of surfaces. Tile self-assembly is a dynamic process based on square *tiles* that have four distinguished sides that can attach to each other under certain conditions. The process starts with a specific predetermined configuration called a *seed* and stops if no further tile can be attached. This model can be implemented in practice using DNA nanotechnology and has been investigated since the late 1990s. As this model has been shown theoretically to allow performing arbitrary computations, it can be seen as a universal computation model. Most works on this topic deal with tile self-assembly in the 2D plane. We study its behaviour on more complex surfaces. The central question that we ask, is, can we design a tile self-assembly system that can detect the type of surface that it is performed on? We do this for two types of surfaces: *flat surfaces* and specific *polycubes*.

Contents

Chapitre 1

Résumé en français

Cette thèse porte sur l'auto-assemblage par tuiles sur des surfaces diverses. En voici un résumé succinct en français, le reste de la thèse étant rédigée en anglais.

De l'assemblage à l'auto-assemblage. L'assemblage de structures complexes à partir de blocs de construction simples liés les uns aux autres est une tâche très générale. Cela se produit, par exemple, lorsque l'on construit des objets complexes, par exemple des machines, à partir de composants plus simples. C'est également une caractéristique fondamentale de la nature : les molécules sont composées d'atomes plus simples. Nous parlons d'*auto-assemblage* lorsque les blocs de construction simples sont dispersés de manière désordonnée et se lient les uns aux autres de manière spontanée et non déterministe pour former une structure organisée.

En fait, les processus naturels sont un exemple fondamental d'auto-assemblage : en effet, la formation de molécules à partir d'atomes peut être considérée comme de l'auto-assemblage. De plus, dans de nombreux processus biologiques, les molécules se lient les unes aux autres pour former des molécules plus grandes, par exemple, les protéines d'ADN et d'ARN qui peuvent être recombinées.

Cela est beaucoup étudié dans le domaine des sciences des matériaux, où l'auto-assemblage est utilisé pour former des cristaux moléculaires [10, 49], des couches moléculaires [74], des molécules médicamenteuses [80], ou d'autres matériaux microscopiques ou nanométriques [83]. Le *Nabat*, une sucrerie persane d'Ispahan, est un exemple d'auto-assemblage de cristaux moléculaires. En Iran, mon pays d'origine, l'une des premières expériences scientifiques pour les élèves des écoles primaires est de fabriquer un cristal de Nabat : préparer une solution sursaturée de sucre, y laisser tremper un fil, et attendre plusieurs jours pour que les cristaux adhèrent au fil. La durée du processus et la forme des cristaux de Nabat dépendent de plusieurs paramètres : la concentration de la solution, la température de l'eau, etc.

Dans cette thèse, nous étudions certains aspects théoriques d'une forme d'auto-assemblage, où les objets de base qui sont assemblés sont des *tuiles*. Ce processus, appelé *auto-assemblage de tuiles*, est très puissant car il permet de réaliser des calculs arbitraires et il est très étudié, à la fois sur le plan théorique et expérimental.

Revêtements par tuiles. Les *tuiles* sont de petits objets relativement plats de différentes couleurs et formes, souvent des carrés. Le revêtement de surfaces a été réalisé depuis l'Antiquité dans l'art et l'architecture.

Les tuiles carrées sont également présentes dans de nombreuses activités de la culture populaire, telles que les jeux de société et les casse-têtes. Ici, les pièces adjacentes doivent avoir des côtés similaires (dans jeux de société, comme les dominos ou le jeu *Carcassonne*) ou complémentaires (comme dans les puzzles).

L'art du revêtement par tuiles a également attiré les mathématiciens : en 1961, Wang [72] a défini les *tuiles de Wang* comme une abstraction mathématique des tuiles : des carrés unitaires avec des bordures spécifiques. La motivation pour de telles tuiles était de savoir si l'on pouvait concevoir des ensembles (non-triviaux) de tuiles qui pourraient être utilisés pour revêtir (sans trous) l'intégralité du plan Euclidien, de telle sorte que si deux tuiles sont placées côte à côte, leurs côtés en contact doivent être identiques. Wang a conjecturé que toutensemble de tuiles permettant un tel revêtement permettrait un revêtement périodique, mais cela a rapidement été réfuté par son étudiant Berger. En 1965 [17], il a montré que ces revêtements pouvaient simuler n'importe quel processus informatique et donc pouvaient être non-périodiques. Il a prouvé l'existence d'un ensemble de 20426 types de tuiles qui pouvaient revêtir le plan, mais seulement de manière apériodique (le plus petit nombre de types de tuiles d'un tel ensemble de tuiles est connu depuis 2021 : 11, voir [41]). Ce résultat a montré l'existence d'une connexion entre les revêtements par tuiles et le calcul : on pourrait, en théorie au moins, construire un ordinateur en utilisant de telles tuiles ! Cependant, pour passer de la théorie à la pratique, il faut passer à des échelles plus petites. En effet, les résultats expérimentaux dans ce domaine, utilisent des nano-tuiles d'ADN.

Informatique basée sur l'ADN et auto-assemblage de nanostructures. Étonnamment, la clé du calcul via les tuiles passe par de minuscules molécules présentes dans tous les êtres vivants. La molécule d'Acide DésoxyriboNucléique (ADN) est une molécule mondialement célèbre composée de deux chaînes (appelées *brins*), liées l'un à l'autre sous la forme d'une double hélice. Cette molécule a été découverte dans les années 1950 par Watson et Crick [73]. Chaque brin d'ADN est formé par une séquence de *nucléotides* (dont il existe quatre types : cytosine [C], guanine [G], adénine [A] et thymine [T]). Les deux brins d'ADN sont liés de telle manière que leurs nucléotides se correspondent deux à deux (A avec T et C avec G), en effet ils forment des liaisons complémentaires. Les molécules d'ADN portent intrinsèquement des informations, et pour la plupart des formes de vie connues sur Terre, l'ADN est essentiel car il encode le développement de l'organisme. Ainsi, on peut le considérer comme le "code source" de la vie.

Inspiré par les processus naturels d'auto-assemblage en chimie et en biologie, dans les années 1980, Ned Seeman a été un pionnier dans la conception de nano-structures basées sur des molécules d'ADN [66]. En effet, deux propriétés de l'ADN sont utiles pour cette tâche. Tout d'abord, les molécules d'ADN se lient les unes aux autres en raison de leurs structures complémentaires come deux pièces de puzzle, donc elles peuvent être utilisées comme blocs de construction pour des structures plus complexes. Deuxièmement, il existe de nombreux types de brins d'ADN d'une longueur donnée (un nombre exponentiel en cette longueur), et donc on peut facilement contrôler quels brins peuvent se lier les uns aux autres. Ainsi, l'ADN est l'ingrédient parfait pour assembler des nano-structures (à condition de pouvoir contrôler de si petites molécules). Seeman a effectivement démontré qu'il était possible de manipuler l'ADN pour construire des structures nanométriques très complexes.

Inspiré par ces travaux sur les nanostructures d'ADN, en 1994, Adelman [2] a conçu un système de calcul nanoscopique basé sur l'ADN résolvant le célèbre problème du *voyageur*

de commerce (TSP) sur une instance avec sept villes. Dans ce problème, on donne un ensemble de villes et leurs distances mutuelles, et il faut trouver une tournée de longueur totale minimale qui passe par toutes les villes. Il s'agit d'un problème très célèbre en informatique théorique, qui est très difficile à résoudre en temps efficace, même pour des ordinateurs puissants. Ainsi, ce résultat a été très inspirant et a démontré le potentiel des techniques de calcul nanométriques basées sur l'ADN.

Auto-assemblage de tuiles. Dans cette thèse, nous étudions l'auto-assemblage de tuiles d'ADN, qui combine les idées du revêtement par tuiles, du calcul d'ADN et de l'auto-assemblage. Ce domaine a été initié par Erik Winfree dans sa thèse de doctorat de 1998 [75]. Inspiré par les travaux de Wang, Seeman, Adelman et d'autres et en combinant leurs approches, il a introduit le *abstract Tile Assembly Model* (aTAM), qui est une traduction mathématique des expériences pratiques réalisées en laboratoire. L'objectif est de concevoir des tuiles (similaires aux tuiles de Wang) avec quelques propriétés supplémentaires, qui peuvent être construites expérimentalement en utilisant des molécules d'ADN. Ces tuiles moléculaires, lorsqu'elles sont placées dans une solution adaptée et en quantités appropriées, s'auto-assemblent et produisent des formes spécifiques qui représentent des calculs.

Les tuiles utilisées dans le modèle aTAM ont quatre côtés spécifiques, appelés *colles*, chacun avec une *force* donnée. Deux tuiles peuvent se *lier* si elles sont adjacentes le long de côtés ayant la même colle. Les tuiles sont placées proches les unes des autres, et elles se lient spontanément de manière non déterministe. Winfree a montré que ce modèle théorique simple peut être utilisé pour calculer, en concevant soigneusement les types de tuiles pour contrôler les assemblages produits. De plus, il a démontré très tôt que l'on peut simuler *n'importe quel processus informatique* réalisé par un ordinateur classique avec ce modèle.

Pour ce faire, certaines tuiles sont pré-assemblées en tant qu'*assemblage initial*. Elles codent l'entrée d'un problème informatique donné (par exemple, deux nombres encodés en codage binaire à l'aide de deux types de tuiles). Ensuite, les tuiles s'assemblent dans n'importe quel ordre, jusqu'à ce qu'aucune autre tuile ne puisse être attachée : nous obtenons un *assemblage terminal*. Cet assemblage représente le résultat du calcul (dans notre exemple, cela pourrait être la somme des deux nombres d'entrée, encore une fois encodée en binaire).

En pratique, de telles tuiles sont construites à l'aide de molécules d'ADN et sont souvent en forme de croix plutôt que de carré [42, 68]. De nos jours, il s'agit d'un domaine de recherche très fructueux, avec de nombreuses applications pratiques et des études expérimentales. Cependant, aller et venir entre les études théoriques et expérimentales dans ce domaine est très difficile.

Questions théoriques en auto-assemblage. Au cours des vingt-cinq dernières années, l'auto-assemblage de tuiles a connu un énorme développement, tant sur le plan théorique que pratique. Du point de vue théorique (qui est celui de cette thèse), il existe de nombreux types de questions pertinentes. Voici quelques exemples de questions qui ont été étudiées dans ce domaine.

– Concevoir des modèles d'auto-assemblage de tuiles qui correspondent aux capacités expérimentales existantes. Comment exprimer aussi fidèlement que possible les contraintes des expériences réelles, tout en conservant la puissance du modèle mathématique ?

– Quels types de formes peuvent être construits par auto-assemblage de tuiles, dans un modèle spécifique ? Peut-on concevoir des systèmes d'auto-assemblage de tuiles qui produisent une catégorie de formes géométriques ? On peut penser ici à des formes comme des carrés, des rectangles, des fractales, etc.
– Concevoir des systèmes d'auto-assemblage de tuiles qui peuvent résoudre des problèmes informatiques spécifiques. Certains problèmes de base qui ont été étudiés sont par exemple de concevoir des compteurs binaires, assembler des formes spécifiées, simuler d'autres modèles de calcul...
– Effectuer l'auto-assemblage aussi efficacement et/ou de manière aussi fiable que possible. Pour l'efficacité, on peut mesurer le nombre de tuiles différentes ou la vitesse de l'assemblage. Pour la fiabilité, on peut limiter le nombre d'erreurs ou demander un seul assemblage terminal possible.
– etc.

Dans cette thèse, nous nous concentrons sur le troisième type de question : notre objectif est de résoudre un type spécifique de problème lié à la surface sous-jacente de l'assemblage, décrit dans la section suivante.

Objectif de la thèse. Le principal objectif de cette thèse est d'étudier l'auto-assemblage de tuiles lorsqu'il est réalisé sur des surfaces qui sont plus complexes que le plan Euclidien 2D. Quels problèmes peuvent être résolus dans ce cadre ? La principale question générale que nous abordons dans cette thèse, est de savoir si l'on peut concevoir des systèmes d'auto-assemblage qui peuvent détecter le type de surface sur laquelle ils se trouvent. Une façon de le faire est de concevoir des systèmes d'auto-assemblage de tuiles qui peuvent être réalisés sur plusieurs types de surfaces, mais qui contiennent des caractéristiques uniques selon le type de surface sur laquelle l'assemblage est effectué.

Nous commençons par cela en utilisant le classique *abstract Tile Assembly Model* (aTAM) sur des *surfaces plates* (en termes intuitifs, des surfaces qui ont une courbure régulière) en considérant quatre types de surfaces planes (de n'importe quelle dimension). Notre stratégie est de concevoir un système d'auto-assemblage de tuiles qui produit des assemblages terminaux qui dépendent du type de surface sur laquelle il est effectué. Pour ce faire, nous définissons certains types spécifiques de tuiles, avec des combinaisons spécifiques de types de tuiles qui apparaissent uniquement dans les assemblages sur un type de surface donné.

Cependant, le modèle aTAM est limité aux surfaces pour lesquelles la notion de direction est cohérente, car les tuiles ne sont pas autorisées à tourner. Cela peut ne pas convenir à certains types de surfaces. Ainsi, l'un de nos objectifs est de proposer un modèle d'auto-assemblage de tuiles qui autorise les rotations de tuiles et qui convient à des surfaces plus complexes. À cette fin, nous introduisons un modèle d'auto-assemblage de tuiles appelé *Surface Flexible Tile Assembly Model* (SFTAM) qui permet d'effectuer l'auto-assemblage de tuiles sur des surfaces quadrangulées complexes, telles que les surfaces de *polycubes* (qui sont des objets 3D constitués de cubes unitaires parallèles aux axes, collés les uns aux autres le long de leurs faces).

D'après les résultats classiques en topologie, on peut dire que la propriété la plus importante d'une surface est son *genre*. Intuitivement, il s'agit du nombre de "trous" présents à l'intérieur de cette surface : une sphère a un genre 0, mais un tore a un genre 1. Si nous effectuons un auto-assemblage sur des surfaces compliquées, nous nous attendons à ce que les formes possibles des assemblages varient fortement en fonction du genre de

la surface sous-jacente. Ainsi, notre objectif principal est de démontrer que nous pouvons utiliser le modèle SFTAM afin de distinguer le genre des polycubes.

À cette fin, nous concevons un système du modèle SFTAM qui permet de détecter le genre d'une classe spéciale de polycubes que nous appelons *cuboides d'ordre 1*. Ces polycubes peuvent avoir un genre 0 ou un genre 1 (dans ce cas, ils ont un tunnel). Notre système a également quelques types de tuiles spéciales qui n'apparaissent dans les assemblages terminaux que si le polycube sous-jacent a un genre 1.

Plan de la thèse. Dans le Chapitre 3, nous commençons par définir les notions mathématiques nécessaires pour cette thèse, telles que les surfaces et leurs propriétés, ou les objets discrets comme les graphes et les polycubes. Nous définissons ensuite formellement l'auto-assemblage par tuiles, en particulier le classique *abstract Tile Asssembly Model* (aTAM). Nous décrivons quelques-unes de ses propriétés de base et les illustrons avec des exemples. Nous passons en revue une partie de la littérature sur l'auto-assemblage de tuiles et les sujets connexes, que nous jugeons pertinents pour cette thèse. En particulier, l'auto-assemblage qui est effectué sur d'autres surfaces que le plan, et les modèles d'auto-assemblage qui assemblent des surfaces 3D, puisque cette thèse étudie l'auto-assemblage sur diverses surfaces.

Dans le Chapitre 4, nous étudions quatre types de *surfaces plates* : le tore plat (avec deux dimensions finies), le cylindre vertical plat et le cylindre horizontal plat (qui sont infinis le long d'une dimension et finis le long de l'autre), et le plan $\mathbb{Z}^2$ (infini dans les deux dimensions). Nous présentons un système d'auto-assemblage par tuiles qui, sur chacune de ces quatre types de surfaces, produit des assemblages terminaux uniques. Nous présentons des constructions pour ces quatre types de surfaces. Ces résultats montrent que l'on peut effectivement détecter le type de surface sur laquelle l'auto-assemblage est effectué, en utilisant le modèle aTAM.

Dans le Chapitre 5, notre objectif est d'effectuer l'auto-assemblage sur des surfaces plus complexes, qui ne sont pas nécessairement aussi régulières que les surfaces plates. Pour ce faire, nous introduisons le nouveau *Surface Flexible Tile Assembly Model* (SFTAM). Ce modèle est conçu pour étendre le modèle aTAM et permettre d'effectuer l'auto-assemblage de tuiles sur des *polycubes*, qui sont des surfaces d'objets 3D obtenus en collant des cubes unitaires parallèles aux axes le long de leurs faces. La raison pour laquelle on ne peut pas utiliser le modèle aTAM à cette fin est que ces surfaces peuvent être très irrégulières, avec divers types de coins, de puits, de tunnels ou de concavités. Ainsi, il est nécessaire d'ajouter la capacité pour les tuiles de tourner (ce qui n'est pas le cas dans le modèle aTAM). Nous comparons le modèle SFTAM avec le modèle aTAM et avec un autre modèle de la littérature, le *Flexible Tile Assembly Model* (FTAM).

Dans le Chapitre 6, nous prouvons quelques résultats techniques sur la façon de concevoir des systèmes d'auto-assemblage de tuiles dans les modèles aTAM et SFTAM pour trouver le milieu d'une *piste* donnée. En termes informels, une piste est une portion d'une surface quadrangulée qui est isomorphe à une grille rectangulaire en deux dimensions. Les résultats de ce chapitre s'appliquent de manière indéterminée au modèle SFTAM pour les pistes sur des surfaces 3D telles que les polycubes, ou dans le modèle aTAM en deux dimensions. Le système qui permet de trouver le milieu d'une piste est appelé *middle finding system*, et notre construction est basée sur des compteurs binaires qui permettent de mesurer la longueur de la piste. Le système de recherche du milieu est utilisé comme bloc de construction dans le chapitre suivant.

Dans le Chapitre 7, nous présentons le principal résultat de cette thèse. Notre objectif

est de montrer que, dans le modèle SFTAM, on peut distinguer entre différents types de surfaces selon leur caractéristique la plus fondamentale : le genre. Comme les polycubes généraux sont difficiles à manipuler et peuvent avoir un genre arbitrairement élevé, nous nous limitons à une famille naturelle de polycubes qui peuvent avoir un genre 0 ou un genre 1, que nous appelons *cuboïdes d'ordre 1*. Intuitivement, ce sont des polycubes qui peuvent être construits à partir de deux cuboïdes (c'est-à-dire des polycubes avec exactement six faces) en soustrayant l'un à l'autre. Nous prouvons un résultat analogue à celui des surfaces plates du Chapitre 4. Plus précisément, nous concevons un système d'auto-assemblage par tuiles dans le modèle SFTAM qui est capable de distinguer le genre du cuboïde d'ordre 1 sur lequel il est effectué. Pour ce faire, le système contient un sous-ensemble spécial Y de types de tuiles, qui apparaît sur chaque assemblage terminal si le cuboïde d'ordre 1 est de genre 1, et aucune tuile de Y n'apparaît jamais si le genre est 0.

Enfin, nous concluons dans le Chapitre 8 en réfléchissant à nos résultats et en proposant d'autres perspectives.

Chapter 2

Introduction

This thesis deals with the subject of tile self-assembly on various surfaces. In this chapter, we briefly introduce the topic.

2.1 From assembly to self-assembly

Assembling complex structures from simple building blocks that are bound to each other is a very general task. It happens for example when humans build complex objects and machines out of simpler components. It is also a basic feature of nature: molecules are composed of simpler atoms. We speak of *self-assembly* when the simple building blocks lie around in a disordered way, and bind with each other in a spontaneous and nondeterministic fashion, to form an organized structure.

In fact, natural processes are a fundamental example of this topic: indeed, the formation of molecules out of atoms can be seen as self-assembly. Even more, in many biological processes, molecules bind with each other to form larger molecules, for example, DNA and RNA proteins that can get recombined.

This is studied a lot in the field of material sciences, where self-assembly is used to form molecular crystals [10, 49], molecular layers [74], drug molecules [80], or other microscopic or nanometric materials [83]. *Nabat*, a Persian sweet from Isfahan, is an example of self-assembly of molecular crystals. In Iran, my home country, one of the

first scientific experiments for students in primary schools is to make a Nabat crystal: preparing a solution supersaturated by sugar, and waiting several days so that crystals stick to the string, as in Figure 2.1. Notably, the duration of the process and the shape of Nabat crystals depends on several parameters: the concentration of the solution, the water temperature, and some other environmental variables of the experiment.

(a) Some Nabat.

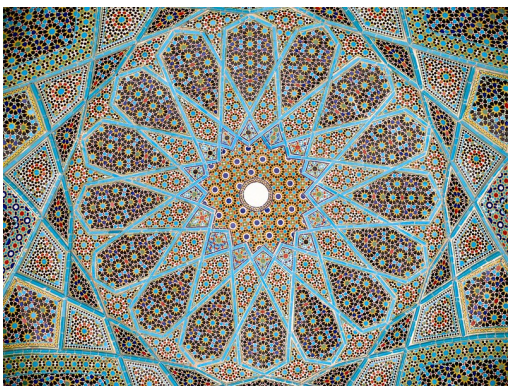
(b) The process of making Nabat crystal.

Figure 2.1: Nabat, the Persian sugar crystals, an example of self-assembly.[1]

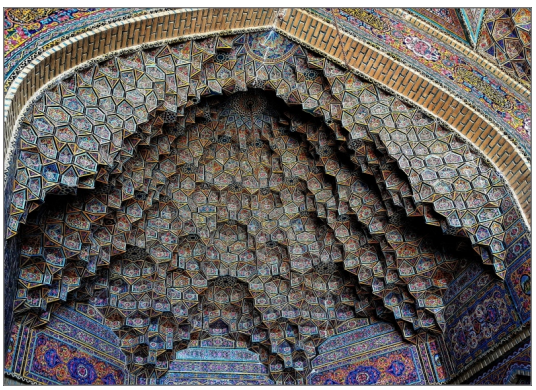
In this thesis, we study some theoretical aspects of a form of self-assembly, where the basic objects that get assembled are *tiles*. This process, called *tile self-assembly*, is very powerful, as it enables to make arbitrary computations, and it is successfully studied both in theory and experimentally.

2.2 Tilings

Tiles are small, relatively flat objects of various colours and shapes, often squares. Tiling surfaces has been performed since antique times in art and architecture, see for example traditional Persian tilings in Figure 2.2.



(a) On a simple surface.



(b) On a complex surface.

Figure 2.2: Traditional Persian tiling.[2]

[1] Images from `https://www.irandestination.com/visit-iran-8/` and `http://aradseyyedtahery.blogfa.com/post/1`.

[2] Images from `https://commons.wikimedia.org/wiki/File:Roof_hafez_tomb.jpg` and `https://atticus.aminus3.com/image/2018-04-18.html`.

Square tiles are also present in many popular culture activities such as board games and puzzles, see Figure 2.3. Here, the adjacent pieces must either have similar or complementary sides.

(a) A jigsaw puzzle.

(b) The game of Carcassonne, where tiles have to be placed while matching the neighboring tiles.

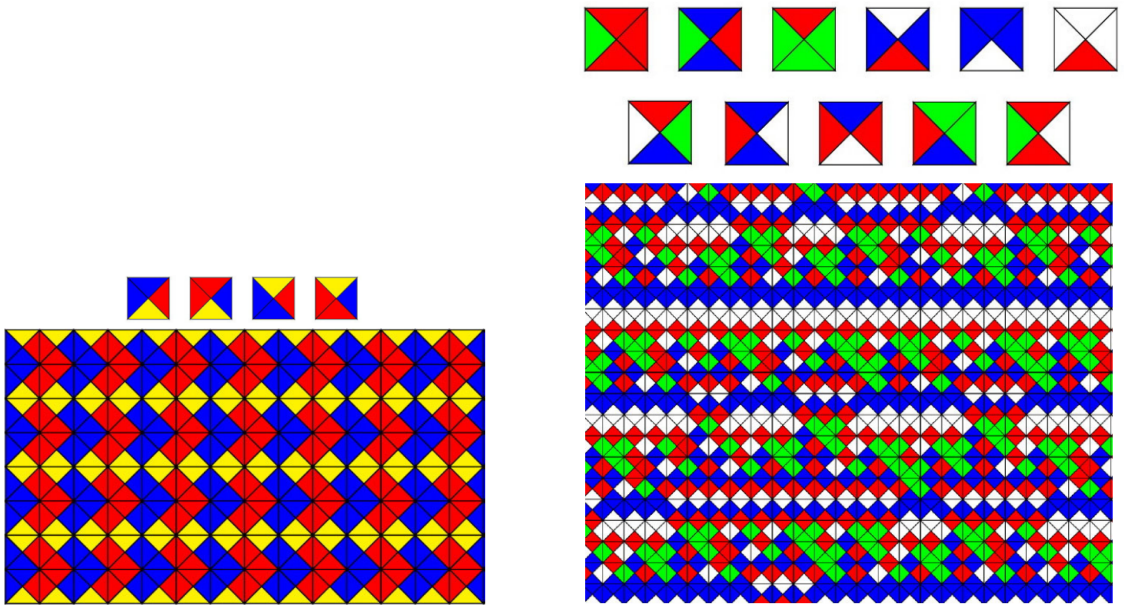
Figure 2.3: Two games based on square tiles that bind at complementary edges (a) or similar edges (b).[3]

The art of tiling has also attracted mathematicians: in 1961, Wang [72] defined *Wang tiles* as a mathematical abstraction of tiles: unit squares with specific borders. The motivation for such tiles was whether one can design sets of tiles that can be used to tile non-trivially and without holes the entire 2D plane, in such a way that if two tiles are placed next to each other, their touching sides must be the same. See Figure 2.4 for an example of such a tiling. Wang conjectured that any tile set that allows such a tiling would allow a periodic one, but this was soon disproved by his student Berger. In 1965 [17], he proved that these tilings could simulate any computational process, and thus, could be non-periodic. He proved the existence of a set of 20426 tile types that could tile the plane, but only in an aperiodic way (the smallest number of tile types of such a tile set is now known to be 11, see [41]). This result showed the existence of a connection between tiling and computation: one could, in theory at least, build a computer using such tiles! But, interestingly, to go from theory to practice, one needs to go to smaller scales. Indeed, experimental results involve nanotiles of DNA.

2.3 DNA computing and self-assembly of nanostructures

Surprisingly, the key to computing via tiles is through tiny molecules that are present in all our bodies. The molecule DeoxyriboNucleic Acid (DNA) is a world-famous molecule composed of two chains (called *strands*), tied to each other in a shape of a double helix: see Figure 2.5. This molecule was fully uncovered in the 1950s by Watson and Crick [73]. Each strand of DNA is formed by a sequence of *nucleotides* (of which there are four types: cytosine [C], guanine [G], adenine [A] and thymine [T]). The two strands are connected in a way that their nucleotides pairwise match (A with T and C with G), indeed they form

[3]Images from `https://commons.wikimedia.org/wiki/File:Editing_FilePuzzle1_found_bw.jpg` and `https://missgeeky.com/2009/03/18/game-review-carcassonne/`.



(a) A periodic tiling using four types of Wang tiles, from [28].

(b) An aperiodic tiling using eleven types of Wang tiles, from [41].

Figure 2.4: Two tilings of the plane using Wang tiles.

complementary bonds. DNA molecules inherently carry information, and for most known lifeforms on Earth, DNA is essential as it encodes the development of the organism. Thus, it can be seen as the "source code" of life.

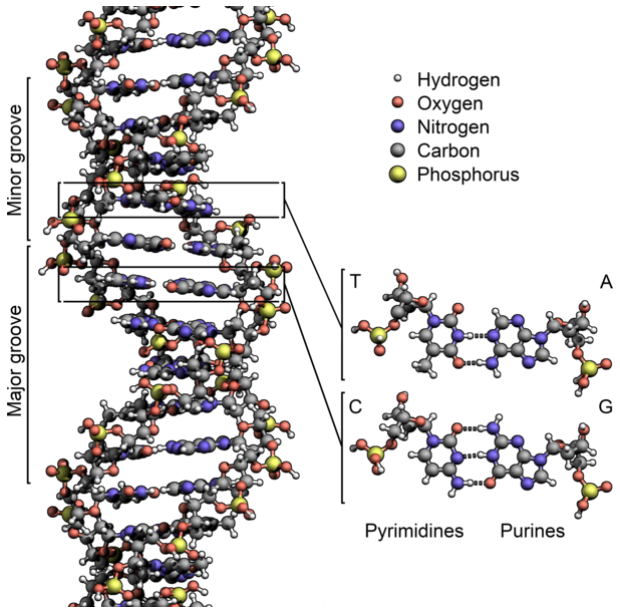


Figure 2.5: The molecular structure of DNA.[4]

Inspired by the natural self-assembly processes in chemistry and in biology, in the 1980s, Ned Seeman pioneered the design of nanostructures based on DNA molecules [66]. Indeed, two properties of DNA are useful for this task. First, DNA molecules bind with

[4]Image from `https://commons.wikimedia.org/wiki/File:DNA_Structure%2BKey%2BLabelled.pn_NoBB.png`

each other (like pieces of a jigsaw puzzle) due to their complementary structures, hence they can be used as building blocks for more complex structures. Second, there are exponentially many types of DNA strands of a given length, and thus one can easily control which strains can bind with each other or not. Thus, DNA is the perfect ingredient for assembling nanostructures (provided one can control such small molecules). Seeman demonstrated indeed that one can manipulate DNA to build very complex nanometric structures.

Inspired by these works on DNA nanostructures, in 1994, Adelman [2] designed a nanoscopic DNA-based computing system solving the *Travelling Salesman Problem (TSP)* on an instance with seven cities. In this problem, one is given a set of cities and their pairwise distances, and one must find a tour with shortest total length that passes through all cities. This is a very famous problem in theoretical computer science that is very difficult to solve even for powerful computers. So, this result was very inspiring and demonstrated the potential of nanometric DNA-based computing techniques.

2.4 Tile self-assembly

In this thesis, we study DNA tile self-assembly, which combines the ideas of tiling, of DNA computing and of self-assembly. This field was pioneered by Winfree in his 1998 PhD thesis [75]. Inspired by the works of Wang, Seeman, Adelman and others and combining their approaches, he introduced the *abstract Tile Assembly Model (aTAM)*, that is a mathematical translation of real-life experiments. The goal is to design tiles (similar to Wang tiles) with some additional properties, that can be built experimentally using DNA molecules. These molecular tiles, when placed in a suitable solution in suitable quantities, self-assemble, and produce specific shapes that represent computations.

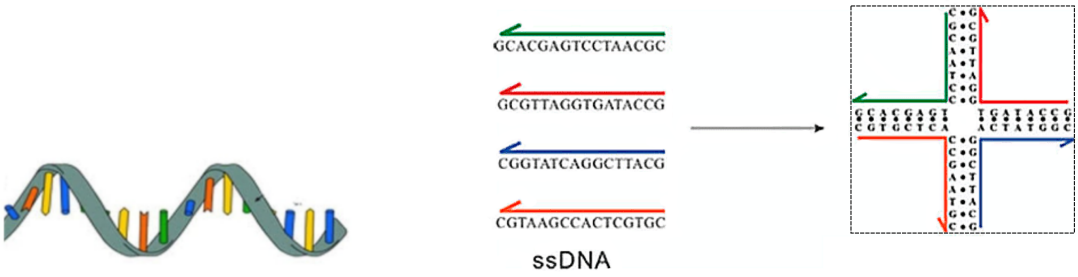
The tiles that are used in the aTAM model have four specified sides, called *glues*, each coming with a specific *strength*. Two tiles can *bind* if they are adjacent along sides that have the same glue. Tiles are placed near to each other, and they bind spontaneously in a nondeterministic fashion. It was shown by Winfree that this simple theoretical model can be used to compute, by carefully designing the types of tiles to control the assemblies. Even more so, it was shown early on that it can simulate *any computational process* performed by a classic computer.

To do this some tiles are pre-assembled as a *seed assembly*. They encode the input of a given problem (for example, two numbers encoded in binary using two types of tiles). Then, tiles assemble in any order, until no further tile can be attached: we obtain a *terminal assembly*. This resulting assembly represents the output of the computation (in our example, it could be the sum of the two input numbers, again encoded in binary).

In practice, such tiles are built using DNA molecules and are often cross-shaped instead of square-shaped [42, 68] (see Figure 2.6 and Figure 2.7). Nowadays, this is a very fruitful research area, with many practical applications and experimental studies. However, going back and forth between the theoretical and the experimental studies in this area is very challenging.

2.5 Theoretical questions in self-assembly

In the last twenty-five years, tile self-assembly has seen a huge development, both in theory and in practice. From the theoretical perspective (which is the one of this thesis),



(a) Single-stranded DNA, from [25].

(b) Schematics of a DNA-based Wang tile, from [69].

Figure 2.6: How to build Wang tiles using DNA.

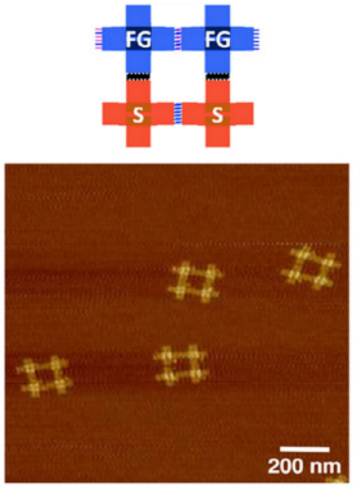


Figure 2.7: An assembly of four cross-shaped DNA-based tiles, from [84].

there are many types of relevant questions. Here are a few examples of questions that have been studied in this field.

- Design tile self-assembly models that correspond to existing experimental capabilities. How to express as closely as possible the constraints from real-life experiments, while keeping the power of the mathematical model?
- What types of shapes can be built by tile self-assembly, in a specific model? Can we design tile self-assembly systems that assemble into a large class of (geometric) shapes? Here one can think of shapes like squares, rectangles, fractals, etc.
- Design tile self-assembly systems that can solve specific computational problems. Some basic problems that have been studied are for example to design binary counters, assemble specified shapes, simulate other computational models...
- Perform self-assembly as efficiently and/or reliably as possible. For the efficiency, one can measure the number of different tiles or the speed of the assembly. For the reliability, one can bound the number of errors or have a unique possible terminal assembly.
- etc.

In this thesis, we are focused on the third type of question: our goal is to solve a specific type of problem related to the underlying surface of the assembly, described in the next section.

2.6 Goal of the thesis

The main goal of this thesis is to study tile self-assembly that is performed on surfaces that are more complex than the 2D Euclidean plane. What problems can be solved in this setting? The main general question that we address in this thesis is whether one can design self-assembly systems that can detect the type of surface that they are located on. One way to do this is to design tile self-assembly systems that can be performed on several types of surfaces, but that contain unique features depending on which type of surface the assembly is actually performed on.

We first do this using the classic abstract Tile Assembly Model (aTAM) on *flat surfaces* (intuitively speaking, surfaces that have a regular curvature) by considering four types of flat surfaces (of any dimension). Our strategy is to design a tile self-assembly system that produces terminal assemblies that depend on the type of surface that it is performed on. To do so, we will have some specific types of tiles, with specific combinations of tile types that will appear only in the assemblies on a given surface type.

However, the aTAM is limited to surfaces that have a consistent concept for directions, as tiles are not allowed to rotate. This may not be suitable for some types of surfaces. Thus, one of our goals is to propose a tile self-assembly model that allow tile rotations and is suitable for more complicated surfaces. To this end, we introduce a tile self-assembly model called *Surface Flexible Tile Assembly Model* (SFTAM) which enables to perform tile self-assembly on complex quadrangulated surfaces, such as surfaces of *polycubes* (3D objects made of axis-parallel unit cubes glued together along their faces).

Arguably, from classic results in topology, the most important property of a surface is its *genus*. Intuitively, this is the number of "holes" present inside that surface: a sphere has genus 0, but a torus has genus 1. If we perform self-assembly on complicated surfaces, we expect that the possible shapes of the assemblies will strongly vary depending on the genus of the underlying surface. Thus, our main goal is to demonstrate that we can use the SFTAM in order to distinguish the genus of polycubes.

To this end, we design a SFTAM system that enables to detect the genus of a special class of polycubes that we all *order-1 cuboids*. These polycubes can have genus 0 or genus 1 (in this case, they have a tunnel). Our system also has some special tile types that appear in terminal assemblies only if the underlying polycube has genus 1.

2.7 Outline of the thesis

In Chapter 3, we start by defining mathematical notions necessary for this thesis, such as surfaces and their properties and discrete objects like graphs and polycubes. We then formally define tile self-assembly, in particular the classic abstract Tile Assembly Model (aTAM). We describe some basic properties and illustrate with examples. We review some of the literature on tile self-assembly and related topics, that are relevant to this thesis. In particular, self-assembly that is performed on other surfaces than the plane, and self-assembly models that assemble 3D surfaces, since this thesis studies self-assembly on various surfaces.

In Chapter 4, we study four types of *flat surfaces*: the flat torus (finite along two dimensions), the flat vertical cylinder and the flat horizontal cylinder (finite along one dimension), the plane $\mathbb{Z}^2$ (infinite in both dimensions). Our goal is to show that self-assembly performed on one of these flat surfaces, can distinguish the surface it is performed

on. More precisely, we design an aTAM tile self-assembly system with two specific tile types x and y. This system can be used on any of the four types of flat surface, in such a way that in any terminal assembly, the subset of $\{x, y\}$ that appears in the assembly uniquely depends on the type of the surface the assembly is performed on.

In Chapter 5, our aim is to perform self-assembly on more complex surfaces, which are not necessarily as regular as flat surfaces. To do so, we introduce the new *Surface Flexible Tile Assembly Model* (SFTAM). This model is designed to extend the aTAM and enable to perform tile self-assembly on *polycubes*, which are surfaces of 3D objects obtained by gluing axis-parallel unit squares along their faces. The reason one cannot use the aTAM for this purpose is that these surfaces can be very irregular, with various types of corners, pits, tunnels or concavities. Thus, one needs to add the ability for tiles to rotate (which is not there in the aTAM). We compare the SFTAM with the aTAM and with another model from the literature, the *Flexible Tile Assembly Model* (FTAM).

In Chapter 6, we prove some technical results on how to design aTAM or SFTAM tile assembly systems to find the middle of a given *track*. Informally speaking, a track is a portion of a quadrangulated surface that is isomorphic to a rectangular grid in two dimensions. The results from this chapter apply indeterminately to the SFTAM for tracks on 3D surfaces such as polycubes, or to the aTAM in two dimensions. The system that enables to find the middle of a track is called *middle finding system*, and the construction is based on binary counters that enable to measure the length of the track. The middle finding system will be used as a building block in the next chapter.

In Chapter 7, we present the main result of this thesis. Our goal is to show that, in the SFTAM, one can distinguish between different types of surfaces according to their most fundamental characteristic: the genus. As general polycubes are difficult to handle and can have arbitrarily high genus, we restrict ourselves to a natural family of polycubes that can have genus 0 or genus 1, that we call *order-1 cuboids*. Intuitively speaking, they are polycubes that can be built from two cuboids (that is, polycubes with exactly six faces) by subtracting one from the other. We provide an analogous result as that for flat surfaces from Chapter 4. More precisely, we design an SFTAM tile self-assembly system that is able to distinguish the genus of the order-1 cuboid it is performed on. To do so, the system contains a distinguished subset Y of tiles that appears on every terminal assembly if the host order-1 cuboid is of genus 1, and no tile of Y ever appears if the genus is 0.

Finally, we conclude in Chapter 8 by reflecting on our results and proposing further perspectives.

Usage of the PyTAS software Many figures in this thesis were produced using the Python-based Tile Assembly Simulator software developed by Patitz and others since 2017 [57]. In this software, one can specify a tile self-assembly system and simulate the assembly, which is shown in a graphical interface. We have used the software to check the validity of some of our systems. As the software also visualizes the tiles and the assemblies, we have used the produced pictures (after some editing using the GIMP software) to illustrate the assemblies of this thesis, mainly in Chapter 4, Chapter 6 and Chapter 7 (sometimes we have drawn the assemblies "manually", using the TikZ LaTeXpackage).

Chapter 3

Preliminaries and state of the art

In this chapter, we present preliminary notions, definitions and examples necessary for understanding the next chapters. We also briefly survey existing results in the area of tile self-assembly (and some related topics) and mention important references. In particular, we mention the previous works of tile self-assembly on surfaces other than the Euclidean plane, and tile self-assembly for 3D objects.

3.1 Preliminary mathematical definitions

We start with some definitions that will be useful to define the necessary notions for tile self-assembly systems.

3.1.1 Mathematical notation

We denote by $\mathbb{N}$ the set of natural numbers, that is, the set $\{1, 2, 3, \ldots\}$ and by $\mathbb{Z}$, the set of integers: $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$.

For a (partial) function $f : X \dashrightarrow Y$, we denote by $Dom(X)$ the *domain* of X (the elements of X over which f is defined) and by $Im(f)$, the *image* of f (the elements of Y that have a pre-image by f).

3.1.2 Graphs

We now give the definition of a graph, which is a fundamental object in discrete mathematics to represent sets of elements that are in relation with each other, and that will be used at many places in this thesis.

Definition 3.1 (Graph)**.** *A (simple, undirected)* graph $G = (V, E)$ *is a discrete object defined over a (possibly infinite) set* V *of* vertices *and a (possibly infinite) set* E *of* edges*, which are pairs of vertices of* V*. Two vertices of* G *are* adjacent *if they belong to the same edge of* E*.*

The notion of subgraph is defined as follows.

Definition 3.2 (Subgraph)**.** *Given a graph* $G = (V, E)$*, a* subgraph $H = (V', E')$ *of* G *is a graph such that* $V' \subseteq V$ *and* $E' \subseteq E$*. For a set* S *of vertices of* V*, the* subgraph of G induced by S *is the subgraph* $H = (S, E_S)$ *of* G *where* E_S *contains all edges of* E *with two endpoints in* S*.*

We also need to define connectivity notions for graphs.

Definition 3.3 (Graph connectivity)**.** *A graph is* connected *if for any two vertices of* G*, there exists a path in* G *between these two vertices. A* connected component *of a graph* G *is a maximal subgraph of* G *that is connected. (If a graph is connected, it has only one connected component.)*

The following notion will be useful to us, to say if two graphs have the same size and the same structure.

Definition 3.4 (Graph isomorphism)**.** *An* isomorphism *of a graph* G_1 *to a graph* G_2 *is a bijective function* $f : V(G_1) \to V(G_2)$ *such that two vertices* v, w *of* G_1 *are connected by an edge if and only if* $f(v)$ *and* $f(w)$ *are connected by an edge in* G_2*.*

3.1.3 The 2D and 3D infinite lattices

We now define two infinite structures, that can be seen as infinite graphs, which will be the hosts of some of our tile assemblies.

Definition 3.5 (2D lattice)**.** *The* 2-dimensional lattice *(*2D lattice *for short) is the infinite graph with vertex set* $\mathbb{Z}^2$*, with an edge between two vertices* (x_1, y_1) *and* (x_2, y_2) *if (i)* $|x_2 - x_1| = 1$ *and* $y_1 = y_2$ *or (ii)* $|y_2 - y_1| = 1$ *and* $x_1 = x_2$*.*

Definition 3.6 (3D lattice)**.** *The* 3-dimensional lattice *(*3D lattice *for short) is the infinite graph with vertex set* $\mathbb{Z}^3$*, with an edge between two vertices* (x_1, y_1, z_1) *and* (x_2, y_2, z_2) *if (i)* $|x_2 - x_1| = 1$*,* $y_1 = y_2$ *and* $z_1 = z_2$*, (ii)* $|y_2 - y_1| = 1$*,* $x_1 = x_2$ *and* $z_1 = z_2$*, or (iii)* $|z_2 - z_1| = 1$*,* $x_1 = x_2$ *and* $y_1 = y_2$*.*

See Figure 3.1 and Figure 3.2 for illustrations of the 2D and 3D lattices, respectively.

Definition 3.7 (Facet of the 2D and 3D lattices)**.** *A* facet *is a face of the lattice* $\mathbb{Z}^2$ *or* $\mathbb{Z}^3$*, i.e. a unit square whose vertices have integer coordinates.*

We can also work with a finite subgraph of the 2D lattice, defined as follows.

Definition 3.8 (Rectangular grid graph)**.** *The* rectangular grid graph $G_{n,m}$ *is the subgraph of the 2D lattice containing all vertices* (x, y) *with* $0 \leq x \leq n - 1$ *and* $0 \leq y \leq m - 1$*.*

See Figure 3.3 for an illustration of the rectangular grid graph $G_{5,9}$.

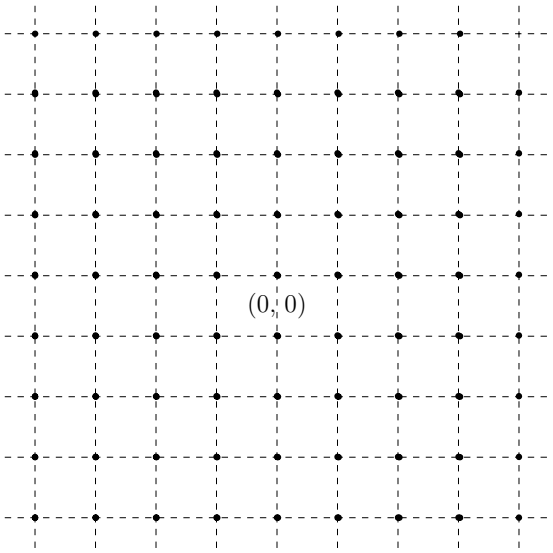


Figure 3.1: The 2D lattice.

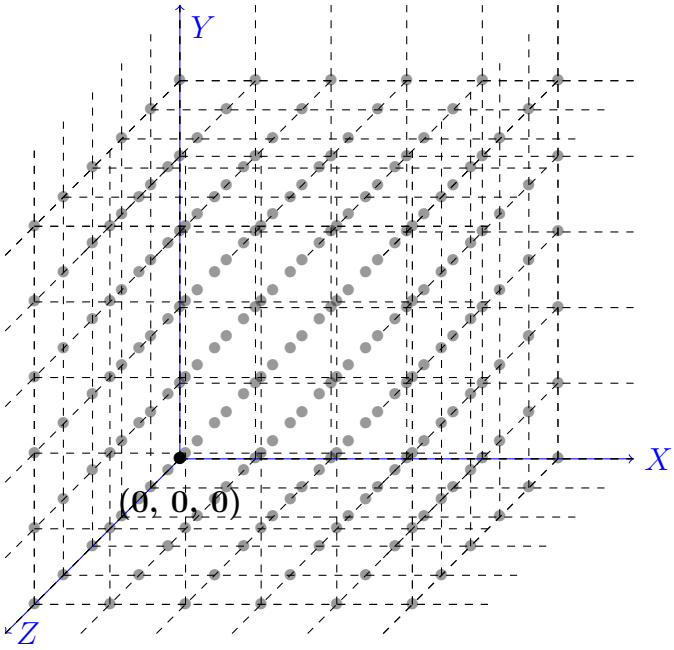


Figure 3.2: The 3D lattice.

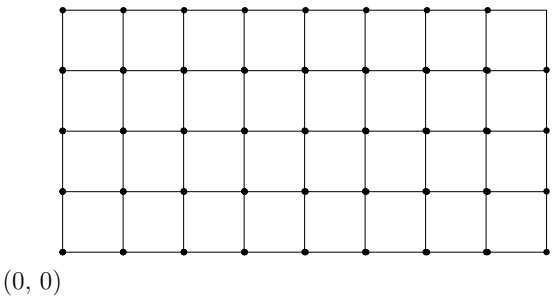


Figure 3.3: The rectangular grid graph $G_{9,5}$.

3.2 Surfaces and associated discrete structures

We now give some definitions about the mathematical objects of surfaces, a part of the field of topology. Indeed, the goal of this thesis is to study tile self-assembly when it is performed on various surfaces. For a deeper mathematical covering of these topics, we refer to the classic textbook [9] on topology and the more recent book on surfaces [56].

3.2.1 Surfaces and discrete surfaces

We begin with some basic definitions.

Definition 3.9 (Topological space)**.** *A* topological space *is a set X whose elements are called points, with a collection τ of subsets of X called* open sets *and satisfying the following axioms:*

- *the empty set and X itself belong to τ;*
- *τ is closed under finite or infinite unions;*
- *τ is closed under finite intersections.*

The following definition is central in topology.

Definition 3.10 (Homeomorphism of topological spaces)**.** *A* homeomorphism *of a topological space S_1 to another topological space S_2 is a bijective and continuous function $f : S_1 \to S_2$ that has a continuous inverse function. If such a homeomorphism exists, we say that S_1 and S_2 are* homeomorphic*.*

Homeomorphisms are the isomorphisms for topological spaces: they preserve all the topological properties of a space. When two topological spaces are homeomorphic, from a topological viewpoint, they are the same.

We are now ready to formally define the notion of a surface.

Definition 3.11 (Surface)**.** *A* (continuous) surface *is a 2-dimensional topological manifold, that is, a Hausdorff topological space such that each point has a neighborhood homeomorphic to an open disk in the Euclidean plane.*

As we will consider discrete versions of surfaces, we need the following definition of a graph embedding.

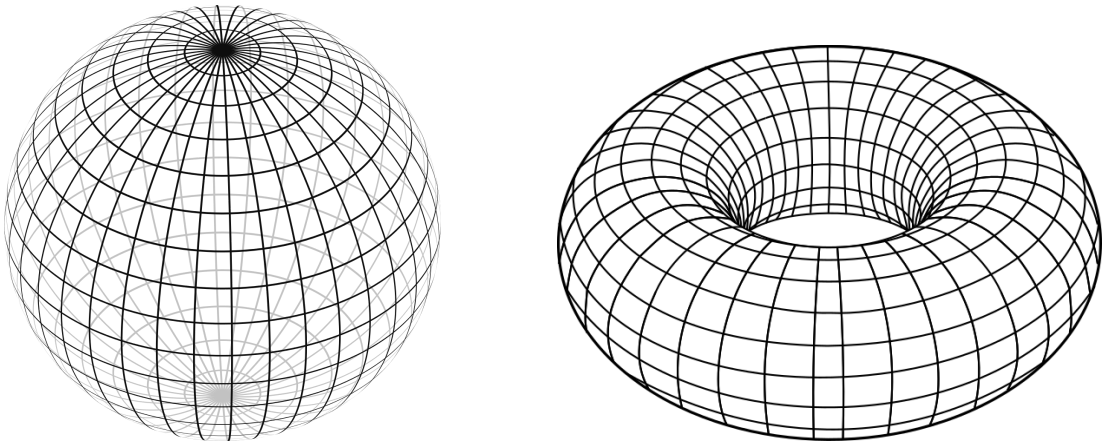
Definition 3.12 (Graph embedding on a surface)**.** *For a graph $G = (V, E)$ and a surface S, an* embedding *of G in S is an injective mapping $\rho_V : V \to S$ of the vertices of G to the points of S together with a mapping $\rho_E : E \to S$ of the edges of E to arcs of S such that any edge xy of G is mapped to a simple arc between $\rho_V(x)$ and $\rho_V(y)$, in a way that any two edges xy and vw may only intersect at their endpoints (in the case where $\{x, y\} \cap \{v, w\} \neq \varnothing$).*

The regions of S that are enclosed by arcs of the embedding are called faces*.*

We can now define discrete surfaces.

Definition 3.13 (Discrete surface)**.** *A* discrete surface *is a surface S with a graph G embedded on S.*

The 2D lattice defined in Definition 3.5 can be seen as a discrete version of the Euclidean plane, partitioned into the set of unit square facets of the 2D lattice. See Figure 3.4 for a discrete sphere and a discrete torus.



(a) A discrete sphere. (b) A discrete torus.

Figure 3.4: Two orientable discrete surfaces.[1]

One may want to restrict the shape of the faces of a discrete surface, as follows.

Definition 3.14 (Triangulation and Quadrangulation)**.** *A* triangulation *of a surface S is a graph embedded on S so that each face is a triangle. A* quadrangulation *of a surface S is a graph embedded on S so that each face is a quadrangle (a polygon with four sides).*

For example, the lattice $\mathbb{Z}^2$, or the surface of a polycube, if seen as graphs, form natural quadrangulations, since their facets are quadrangles. Quadrangulations are naturally useful for tile self-assembly, since tiles are quadrangles. See Figure 3.4(b) for a quadrangulated torus.

Orientability and genus are the most classical tools for categorising surfaces. Here is an informal definition of orientability (for a rigorous, but rather complicated, definition, see the book [56]).

Definition 3.15 (Orientability of surfaces)**.** *A surface is called* orientable *if it admits a consistent and continuous notion of clockwise and counterclockwise orientations at every point of the surface. Otherwise, it is called* non-orientable*.*

The Euclidean plane $\mathbb{Z}^2$, the sphere or the torus are examples of orientable surfaces. A Möbius strip (Figure 3.5) is an example of a non-orientable surface.

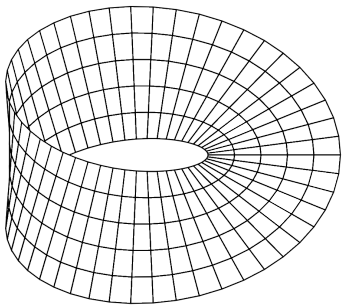


Figure 3.5: A discretized Möbius strip, a simple non-orientable surface.[2]

[1] Images from `https://commons.wikimedia.org/wiki/File:Sphere_wireframe_10deg_3r.svg` and `https://commons.wikimedia.org/wiki/File:Simple_Torus.svg`.

[2] Image from `https://commons.wikimedia.org/w/index.php?curid=15278313`.

3.2.2 Genus of a surface

The most important property of an orientable surface is its *genus*. This notion was defined during the 19th century, when surfaces were studied more deeply than before, see the book [62] for more details, (in particular, the term *genus* was first used by Clebsch in 1865 see [62, p.59]). The genus can be defined as follows.

Definition 3.16 (Genus of a surface)**.** *The* genus *of an orientable surface S is the maximum number of closed curves that one can cut from the surface, while keeping it connected.*

The Euclidean plane and the sphere are examples of orientable surfaces of genus 0. The torus is an example of orientable surface of genus 1. For two polycubes of genus 1 and 2, see Figure 3.6.

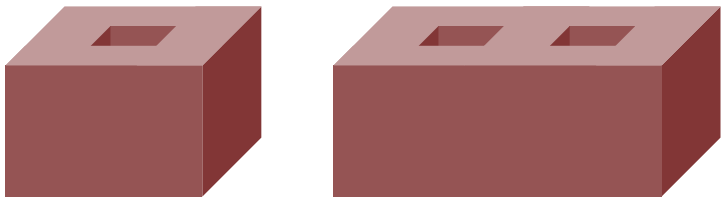


Figure 3.6: Two polycubes, of genus 1 and 2.

The importance of the genus stems from the notion of homeomorphism of surfaces (Definition 3.10). Informally, two surfaces are homeomorphic if we can obtain one from the other by stretching, bending, squeezing or shrinking it (possibly after cutting it to unknot it, and gluing the sides of the cut back together in the exact same place as before). A famous joke is that, supposedly, a doughnut and a coffee mug are both indistinguishable to a topologist, since they are both homeomorphic to a torus: indeed they have the same genus (one), and they are homeomorphic. See Figure 3.7 for a visualization of this homeomorphism.

Figure 3.7: A doughnut and a coffee mug are the same from the viewpoint of a topologist.[3]

[3]Image from `https://commons.wikimedia.org/wiki/File:Topology_joke.jpg`.

The notion of genus is also important for graphs. The definition can be transferred to graphs, through the notion of embedding of a graph on a surface (Definition 3.12).

Definition 3.17 (Genus of a graph)**.** *The* genus *of a graph G is the smallest genus of a surface on which G can be embedded.*

Graphs that can be embedded on a surface of genus 0 are called *planar* (for example, all triangulations of a sphere are planar). Not all graphs are planar, for example, the complete graph on five vertices is a famous example of a graph with genus 1. See the book [39] for more on this topic.

The following definition comes from the works of Euler on polyhedra. In fact, the formula was already proved by Descartes but not published (see [62, p.147]). For an excerpt of the original paper see Figure 3.8.

Definition 3.18 (Euler number, [31])**.** *Let S be a discrete orientable surface coming from a graph G embedded on S, with v vertices, e edges and f faces. The* Euler number *of S is $v - e + f$.*

PROPOSITIO IV.

§. 33. *In omni ſolido hedris planis incluſo aggregatum ex numero angulorum ſolidorum et ex numero hedrarum binario excedit numerum acierum.*

DEMONSTRATIO.

Scilicet ſi ponatur vt hactenus:
numerus angulorum ſolidorum = S
numerus acierum - - - - = A
numerus hedrarum - - - = H
demonſtrandum eſt, eſſe S + H = A + 2.

Figure 3.8: The Euler formula in Euler's original paper [31], in Latin, and reproduced from [62, p.147].

Euler showed that $v - e + f = 2$ for graphs embedded on the discrete sphere, independently of the graph and the embedding. More generally, it is known that the Euler number determines the genus of the surface, as follows (this theorem is attributed to the admiral De Jonquières by Poincaré, see [62, p.150]).[4]

Theorem 3.19 (De Jonquières)**.** *For any discrete orientable surface of genus g, we have $v - e + f = 2 - 2g$.*

The value $v - e + f$ is usually denoted by χ, and it is called the *Euler characteristic* of a discrete surface.

What happens for more general (not necessarily discrete) surfaces? Famous mathematicians like Jordan and Möbius already had the intuition in the 19th century that for all *compact*[5] surfaces, the genus determines the surface up to homeomorphism. After many attempts to show this, it was finally proved by Brahana in 1921 [20], as the *Classification Theorem for Compact Surfaces*. See the book [34] for the history and a presentation of the proof. One main difficulty of the proof is to show that this type of surface can always

[4] Poincaré later showed that such a formula can be extended to dimensions higher than 3.

[5] A surface with point set X is *compact* if, for every collection C of open subsets of it whose union is X, there is a finite subcollection of C whose union is X. For example, the sphere or the torus are compact, but if one removes a finite set of points from one of them, it becomes non-compact.

be triangulated. The following statement is a simplified version of the Classification Theorem for Compact Surfaces (for orientable surfaces without boundaries) and shows the importance of the genus for "natural" surfaces.

Theorem 3.20 (Brahana [20], see also [34, Theorem 6.1])**.** *Two compact orientable surfaces without boundary are homeomorphic if and only if they have the same genus.*

Thus, the genus determines the shape of a surface. Using the homeomorphism between surfaces of the same genus, the geometric constructions that can exist on one surface (such as arrangements of cycles or lines) also exist on any surface of the same genus (after some stretching and resizing).

In the context of tile self-assembly performed on a given surface, these constructions are the shapes of the assemblies. Intuitively, it seems that tile self-assembly performed on two surfaces with the same genus will behave similarly (as compared to surfaces with different genus). This is the reason why the focus of Chapter 7 is on the genus of a given surface.

Perhaps the most famous result in the area of surface classifiation is the 1904 Poincaré conjecture, which was deemed a "millenium problem" by The Clay Mathematics Institute, and which was proved by Perelman in 2003 [51]. It states that every simply connected, closed 3-manifold is homeomorphic to the 3-sphere (thus, this concerns surfaces in 4D, which we do not consider in this thesis).

3.2.3 Polycubes

We now define 3D objects called *polycubes*. Indeed, Chapters 5 and 7 of this thesis deal with tile self-assembly performed on polycube surfaces.

Informally speaking, polycubes are special kinds of orthogonal polyhedra made from unit cubes that are glued together along their faces. See Figure 3.9 for some examples. Polycubes are a natural discrete version of 3D surfaces, indeed for any 3D surface one can approximate it using sufficiently small polycubes [23]. Polycubes have been studied in previous works in the context of self-assembly [18, 19, 30] and origami-like folding processes in 3D [6]. Polycubes generalize polyominoes in the 2D plane [6, 18].

For the formal definition, we work in 3-dimensional space, on the integer lattice $\mathbb{Z}^3$. Recall that a *facet* is a unit square in $\mathbb{Z}^3$ with integer coordinates (Definition 3.7).

Definition 3.21 (Polycube)**.** *A* polycube *is a 3D structure that is a subset of $\mathbb{Z}^3$ and is formed by the union of axis-parallel unit cubes whose vertices have integer coordinates, and that are pairwise attached by their facets.*

All small polycubes that fit inside a 2×2 cube are represented in Figure 3.9.

See Figure 3.10 for a polycube containing 11 facets, 10 of them being on its surface.

It will be useful to define graphs from polycubes as follows.

Definition 3.22 (Facet graph)**.** *Given a polycube C, we define its* facet graph $G_f(C)$ *as the graph whose vertex set is the set of facets on the surface of C, and where two facets are adjacent if they share the same edge of C.*

An example with two simple polycubes and their facet graphs is given in Figure 3.11.

In Figure 3.9, notice the possible complexity in terms of how many facets can be incident with a given vertex. In these polycubes, the number of facets incident with the vertex in the middle of the figure are respectively: 3 in (a), 4 in (b), 5 in (c), 4 in (d), 6

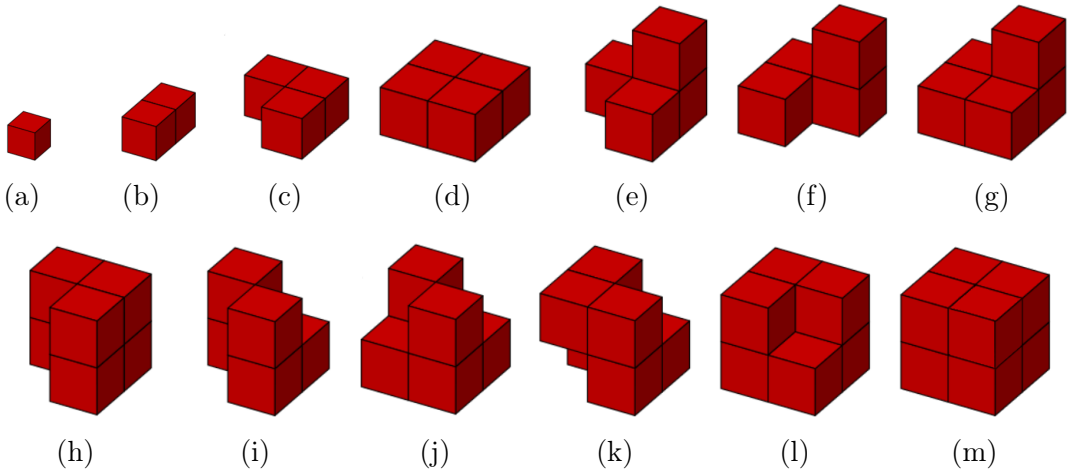


Figure 3.9: All polycubes that fit in a 2×2 cube, from [30].

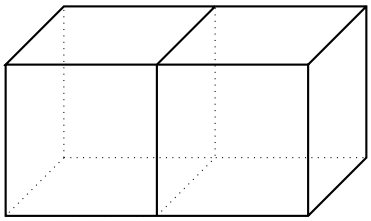
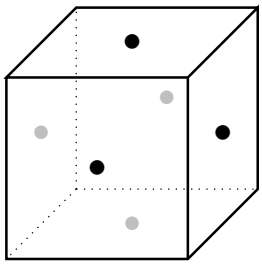
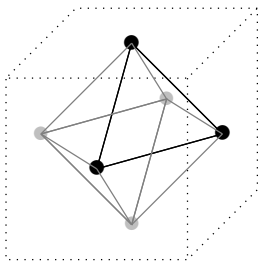


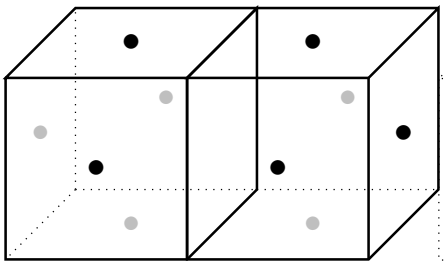
Figure 3.10: A polycube made of two unit cubes. It contains 11 facets, but only 10 of them belong to the surface of the polycube.



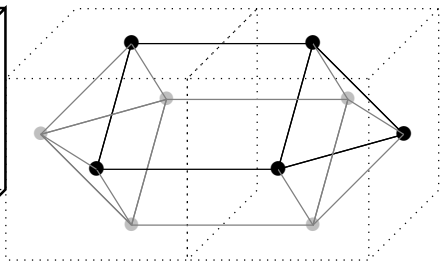
(a) A polycube C_1 made of a single unit cube.



(b) The facet graph of C_1.



(c) A polycube C_2 made of two unit cubes.



(d) The facet graph of C_2.

Figure 3.11: Two polycubes with their corresponding facet graphs.

in (e), 6 in (f), 5 in (g), 4 in (h), 7 in (i), 6 in (j), 6 in (k), 3 in (l), 0 in (m). We can

define different types of vertices of a polycube as follows.

Definition 3.23 (Types of polycube vertices)**.** *Let C be a polycube and v, a vertex of C.*

Vertex v is concave *if there are two unit-length edges vx and vy of C (x, y are vertices of C) containing v forming a right angle, and the segment xy is not in the inside of C.*

On the other hand, vertex v is convex *if for any two unit-length edges vx and vy containing v, the segment xy is in the inside of C.*

For example, all vertices in the polycube of Figure 3.9(a) are convex, while the central vertices of the polycubes of Figure 3.9(e) and Figure 3.9(l) are concave. The central vertex of the polycube in Figure 3.9(d) is neither concave nor convex.

3.3 The abstract Tile Assembly Model (aTAM)

In this section, we formalize the classic model for tile self-assembly, called abstract Tile Assembly Model (aTAM), and we review some results from the literature. Before, we briefly summarize the history of the area.

3.3.1 History

The field of tile self-assembly originates in the area of *tiling*. In 1961, Wang [72] introduced *Wang tiles*, that is, 2-dimensional unit squares with labels/colors on each edge (later called *glues*). Berger [17] proved in 1965 that this model provides a Turing-universal computation model, based on these tiles. More precisely, he proved that for every Turing machine M with some input I, there exists a tile set $T(M, I)$ that can tile the entire plane if and only if M halts on input I.

In the 1980s, Seeman [66, 42] developed the area of *DNA nanotechnology*. The idea of this field is to use the power of DNA molecules (that can attach each other using their complementary patterns) to assemble nanoscopic objects.

In 1994, Adleman [2] used such tiny DNA objects to perform computations. He encoded an instance of the famous *Travelling Salesman Problem* as DNA molecules, and the molecules assembled in such a way as to provide a solution to the problem. This feature is now labelled as *DNA computing*.

In 1998, inspired by Wang tiles and the works of Seeman and Adleman, Winfree introduced in his PhD thesis from 1998 [75], the *abstract Tile Assembly Model* (aTAM), which was refined in the 2001 PhD thesis of Rothemund [63], and by Rothemund and Winfree together in [65]. This model uses Wang tiling, with the extra information of a non-negative integer strength for each glue label, and the notion of *temperature* to help controlling the assemblies. Self-assembly was implemented in practice, and although the aTAM model is a simplification of real-life experiments, many theoretical works and techniques have been confirmed experimentally.

Winfree showed that the power of DNA self-assembly enables to compute anything that (given enough time) is computable by a computer: tile self-assembly is Turing-universal [76]. This has proved the importance of this model and increased he interest of researchers.

After these early results, numerous other works have been performed on tile self-assembly, both on the theoretical side and on the practical side. Many models have been introduced over the years, and many problems have been studied. We will review some of them, however we cannot cover them all here. We refer to the 2014 survey by Patitz [58] for

more details on the literature, and to the online bibliography of Seeman's laboratory [67]. See also the 2008 PhD thesis of Becker [12], the 2014 PhD thesis of Evans [32], the 2015 PhD thesis of Hendricks [40] and the 2022 PhD thesis of Bohlin [18].

3.3.2 Definitions

We now proceed with the formal definitions for tile self-assembly in the aTAM.

Definition 3.24 (Tile type in the aTAM)**.** *Let Σ be a finite label alphabet and ϵ represent the null label. A* tile type *t is a 4-tuple $t = (t_1, t_2, t_3, t_4)$ with $t_i \in \Sigma \cup \{\epsilon\}$ for each $i = \{1, 2, 3, 4\}$. Each copy of a tile type is a* tile*. t_1 is the* northern glue*; t_2 is the* eastern glue*; t_3 is the* southern glue*; t_4 is the* western glue*. A* strength *$str : \Sigma \cup \{\epsilon\} \to \mathbb{N}$ will be given to each label. A* tile *of a given tile type t is a 2D unit square whose sides are assigned the glues of t.*

A glue label of ϵ indicates that the tile cannot bind on this side of the tile.

Tiles are allowed to translate, but not to rotate. Figure 3.12 shows a representation of a tile of type $t = (a, b, c, d)$ with $str(a) = 2$ and $str(b) = str(c) = str(d) = 1$. The glue labels are written near the corresponding edges of the tile (glues with label ϵ are usually omitted). We represent the strength of the glue by either one or two notches (in this thesis all considered strengths are at most 2).

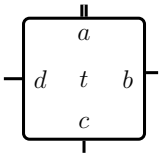


Figure 3.12: An aTAM tile of type $t = (a, b, c, d)$ with $str(a) = 2$ and $str(b) = str(c) = str(d) = 1$.

In the definition of tile types, we show labels by numbers rather than cardinal directions. In Figure 3.12, the northern label is a, the eastern label is b, the southern label is c, and the western label is d.

A *position* (for placing a tile) is a couple (x, y) of $\mathbb{Z}^2$.

Now we can define a *tile assembly system* in the aTAM.

Definition 3.25 (Tile assembly system (TAS) in the aTAM)**.** *A* tile assembly system*, or TAS, in $\mathbb{Z}^2$ is a quintuple $\mathcal{S} = (\Sigma, T, \alpha_\sigma, str, \tau)$, where:*

- *Σ is an alphabet;*
- *T is a finite set of tile types on Σ;*
- *the* seed assembly *(or* seed *for short) α_σ is a set of tiles, each associated with a distinct position of $\mathbb{Z}^2$;*
- *str is a function from $\Sigma \cup \{\epsilon\}$ to non-negative integers called* strength function *such that $str(\epsilon) = 0$; and*
- *$\tau \in \mathbb{N}$ is called the* temperature*.*

We now give a first example of a TAS in the aTAM. This is a very simple TAS, with three tile types.

Example 3.26 (aTAM TAS)**.** *Let $\mathcal{S} = (\Sigma, T, \alpha_\sigma, str, \tau)$ be the aTAM TAS where:*

- *$\Sigma = \{a, b, c\}$ is the alphabet;*

- $T = \{t_0, t_1, t_2\}$ *with* $t_0 = (a, a, b, c)$, $t_1 = (b, \epsilon, \epsilon, a)$ *and* $t_2 = (\epsilon, c, a, \epsilon)$ *are the tile types;*
- *the seed assembly* α_σ *is a tile of type* t_0 *placed at* $(0, 0)$*;*
- $str(\epsilon) = 0$, $str(a) = 1$ *and* $str(b) = str(c) = 2$*;*
- *the temperature* $\tau = 2$.

The tile types of the TAS $\mathcal{S}$ from Example 3.26 are depicted in Figure 3.13; for a better visualization, we give a distinct color to each tile type and to the different glues.

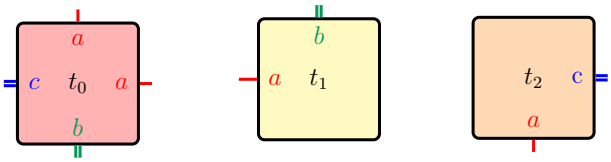


Figure 3.13: The tile types of the aTAM TAS $\mathcal{S}$ from Example 3.26.

Remark 3.27. *Although in Definition 3.25, a TAS is defined with a unique seed assembly, in practice, the seed assembly is often regarded as the input of the TAS, which behaves differently depending on that input. Thus, we often relax this by describing, for the same TAS, a family of allowed seed assemblies.*

Next, we present some further notations and definitions regarding the assembly dynamics. Note that tiles can be placed anywhere on $\mathbb{Z}^2$, but cannot rotate.

Definition 3.28 (Assembly)**.** *An* assembly α *of an aTAM TAS* $\mathcal{S} = (\Sigma, T, \alpha_\sigma, str, \tau)$ *on* $\mathbb{Z}^2$ *is a partial function* $\alpha : \mathbb{Z}^2 \dashrightarrow T$.

For two adjacent positions (x_1, y_1) *and* (x_2, y_2) *of* $\mathbb{Z}^2$ *with* $\alpha(x_1, y_1) = t$ *and* $\alpha(x_2, y_2) = t'$*, we say that* t *and* t' bind *if the glues of* t *and* t' *on the touching sides of* t *and* t' *are equal and have positive strength.*

Definition 3.29 (Assembly graph)**.** *The* assembly graph G_α *associated with an assembly* α *has the set* $Im(\alpha)$ *as its vertices, and two positions* (x_1, y_1) *and* (x_2, y_2) *are adjacent in* G_α *if the tiles* $\alpha(x_1, y_1)$ *and* $\alpha(x_2, y_2)$ *bind.*

See Figure 3.14 for the assembly graph of the assembly from Figure 3.19(g).

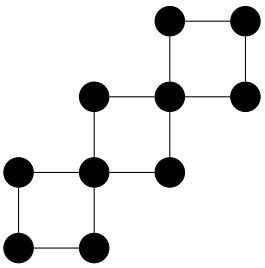


Figure 3.14: The assembly graph of the assembly from Figure 3.19(g).

Definition 3.30 (τ-stable assembly)**.** *For an integer* τ *(usually, the temperature of the TAS), an assembly* α *is* τ-stable *if for breaking the assembly graph* G_α *into several connected components by removing any set* C *of edges of* G_α*, the sum of strengths of the bonds corresponding to edges of* C *is at least* τ.

Two simple assemblies with different values of stability are given in Figure 3.15. In the left assembly, the two tiles are attached via a glue of strength 1, so the assembly is

1-stable but not 2-stable. In the right assembly however, since the tiles are attached by a glue of strength 2, the assembly is 2-stable.

(a) A 1-stable assembly. (b) A 2-stable assembly.

Figure 3.15: Two assemblies of the TAS $\mathcal{S}$ from Example 2.26. With temperature $\tau = 2$, the left assembly is not producible, but the right assembly is producible.

In the process of tile self-assembly, we start with the seed assembly and then we add tiles one by one. This is formally described as follows.

Definition 3.31 (Producible and terminal assemblies)**.** *Let $\mathcal{S} = (\Sigma, T, \alpha_\sigma, str, \tau)$ be an aTAM TAS. An assembly α of $\mathcal{S}$ is* producible *if either α is α_σ, or if α can be obtained from a producible assembly β by τ-stably adding a single tile from T.*

We write $\beta \to^{\mathcal{S}} \alpha$ when α is producible from β and we denote the set of producible assemblies of $\mathcal{S}$ by $A[\mathcal{S}]$.

A producible assembly α is the result $res(\alpha)$ *of an* assembly sequence $\alpha_0 \to \alpha_1 \to \ldots \to \alpha$, *where $\alpha_0 = \alpha_\sigma$.*

An assembly is terminal *if no tile can be τ-stably attached to it. The set of producible, terminal assemblies of $\mathcal{S}$ is denoted by $A_\square[\mathcal{S}]$.*

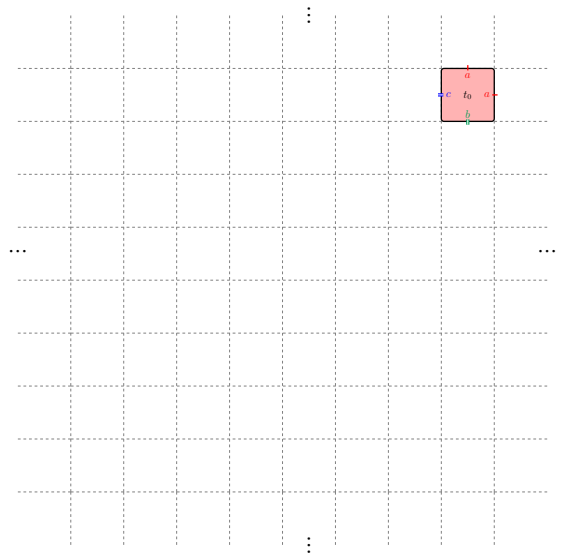
As an example, a more interesting producible assembly is shown in Figures 3.16, 3.17, 3.18 and 3.19. The assembly starts with a single seed tile of type t_0. On each of its south and west glues, which have strength 2, tiles of type t_1 and t_2, respectively, can attach. As these tiles each have a strength 1 glue at the west and south, respectively, since the temperature τ is 2, this creates room for another tile of type t_0 to attach to these two tiles simultaneously. Then, the assembly can go on indefinitely in this way, creating a ribbon directed from north-east to south-west.

Remark 3.32. *We note that aTAM tile self-assemblies are inherently nondeterministic and asynchronous. Thus, one of the challenges in designing a TAS in the aTAM is to overcome this limitation to control the growth of the assembly as required.*

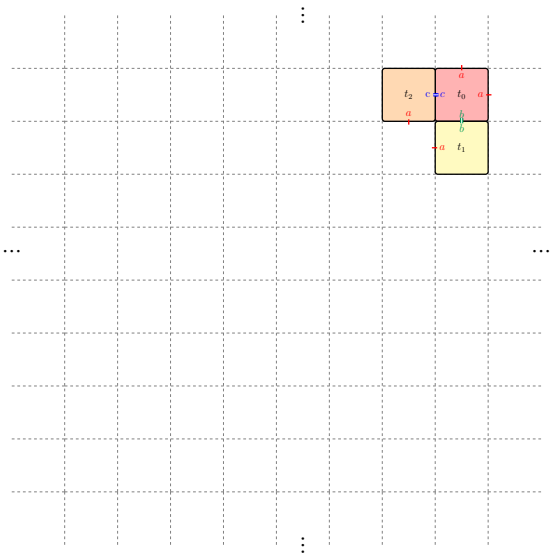
As $\mathbb{Z}^2$ is infinite, one can also consider infinite assemblies in the aTAM. Most definitions from the classic aTAM transfer to the infinite case. We will need the following definition for infinite assemblies, that slightly vary, as an infinite assembly can be called terminal.

Definition 3.33 (Infinite assemblies)**.** *The* result *of an infinite assembly sequence is its* limiting assembly*. An infinite assembly is* terminal *if no tile can be τ-stably attached to it.*

Infinite self-assembly is studied mainly to uncover the absolute limits of the model. In particular, what kind of infinite shapes can or cannot be assembled, under various constraints on the type of assembly. We refer to the survey [59] for more on infinite aTAM assemblies.

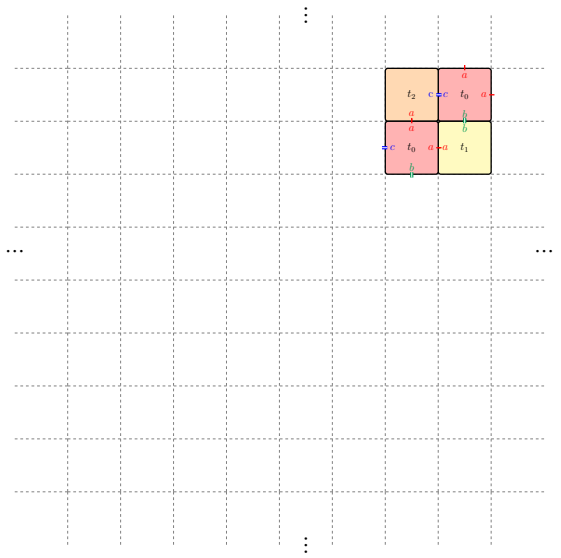


(a) First stage of the assembly: the seed is placed.

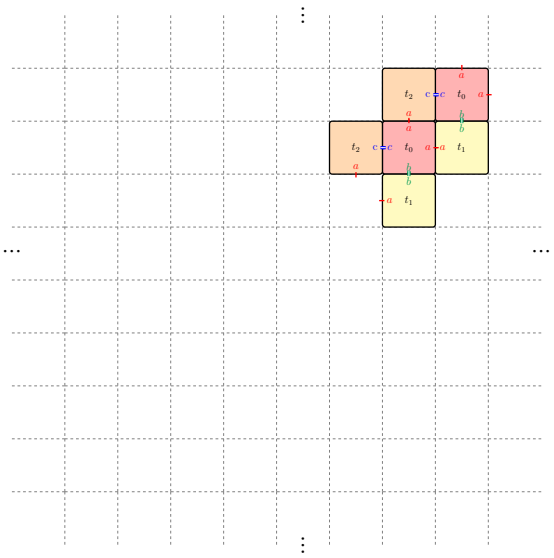


(b) Second stage of the assembly, after addition of two tiles of type t_1 and t_2.

Figure 3.16: Example aTAM assembly for the TAS in Example 3.26 (part 1 of 4).



(c) Third stage of the assembly: one tile of type t_0 is added.



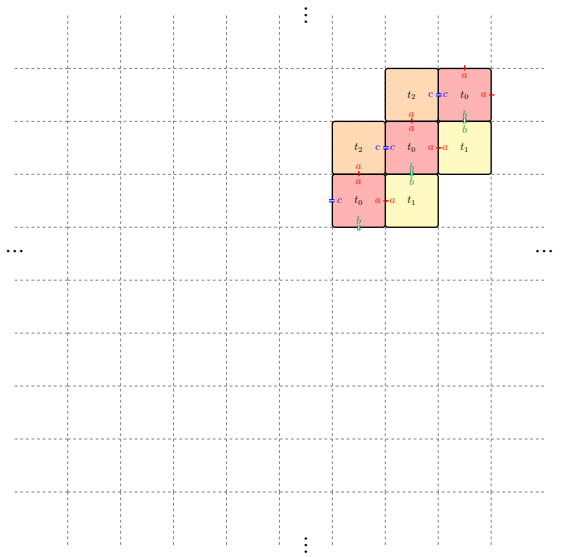
(d) Fourth stage of the assembly: two tiles of type t_1 and t_2 are added.

Figure 3.17: Example aTAM assembly for the TAS in Example 3.26 (part 2 of 4).

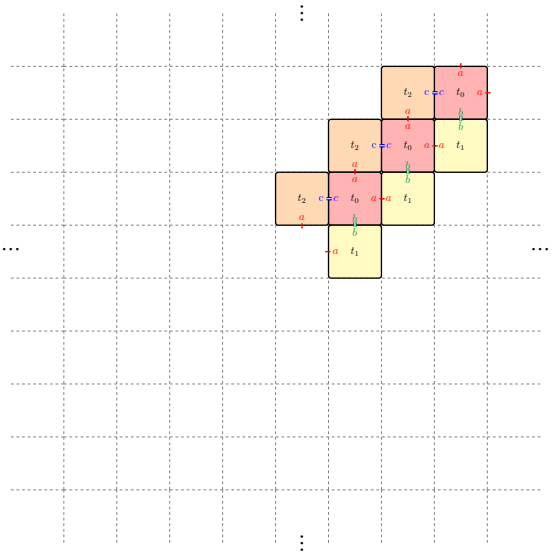
3.3.3 Fundamental examples: simulating counters via the aTAM

One fundamental ingredient of many complex tile self-assembly systems are binary counters, as they enable to control the growth of assemblies, as well as complex computations.

We first present an aTAM TAS that represents a limited counter. It is able to determine the parity of the number of tiles labelled "one" in an input seed. It is taken from [24] (with the minor modification of the final seed assembly tile, in order to obtain terminal assemblies).

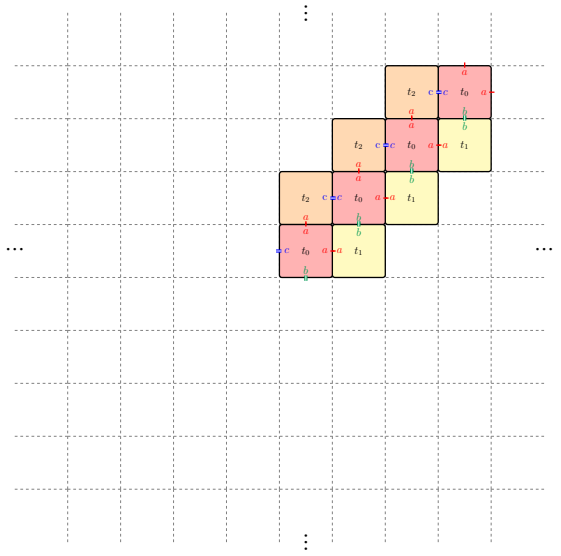


(e) Fifth stage of the assembly: one tile of type t_0 is added.

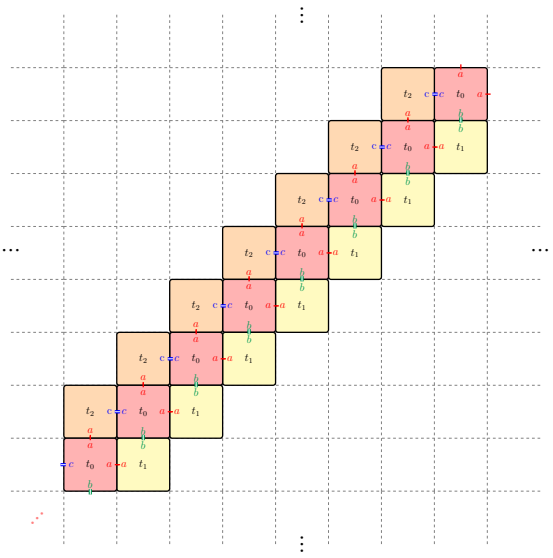


(f) Sixth stage of the assembly: two additional tiles of types t_1 and t_2 are placed.

Figure 3.18: Example aTAM assembly for the TAS in Example 3.26 (part 3 of 4).



(g) Seventh stage of the assembly.



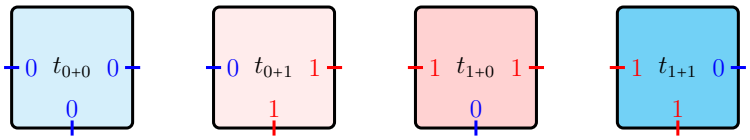
(h) Advanced stage of the assembly.

Figure 3.19: Example aTAM assembly for the TAS in Example 3.26 (part 4 of 4).

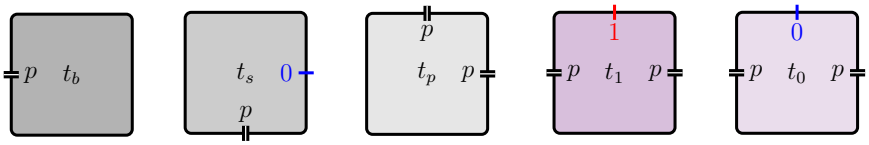
Example 3.34 (Parity aTAM TAS [24]). *Let* $\mathcal{S}_p = (\Sigma, T, \alpha_\sigma, str, \tau)$ *be the TAS where:*

- $\Sigma = \{0, 1, p\}$ *is the alphabet;*
- $T = \{t_{0+0}, t_{0+1}, t_{1+0}, t_{1+1}, t_b, t_s, t_p, t_0, t_1\}$ *(see Figure 3.20 for the tile types);*
- *the seed assembly* α_σ *is composed of a tile of type* t_p *at* $(0,0)$*, a tile of type* t_s *at* $(0,1)$*, a sequence of* k *tiles, each of type* t_0 *or* t_1 *at* $(1,0), \ldots, (k,0)$*, and a tile of type* t_b *at* $(k+1, 0)$*.*
- $str(\epsilon) = 0$*,* $str(0) = str(1) = 1$ *and* $str(p) = 2$*;*
- *the temperature* $\tau = 2$*.*

The tile types of the TAS $\mathcal{S}_p$ from Example 3.34 are depicted in Figure 3.20. We have a seed that is made of a horizontal row of tiles representing zeroes (type t_0) and ones (type t_1). The goal is to have a terminal self-assembly that puts at the rightmost tile, a tile representing the parity of number of ones that have been placed in the seed assembly: if there is an even number of "one" tiles, the eastern glue label will be "0", and otherwise it will be "1". Two terminal assemblies are given in Figures 3.21 and 3.22. In Figure 3.21, the seed represents the input binary string "11001". Since the temperature is 2, the only position where a tile can be placed is at the leftmost side of the row above the seed. Since the leftmost bit tile of the seed is of type t_1, the only tile that can attach is of type t_{0+1}, indicating that at this stage, the number of detected tiles of type t_1 was even (in this case, zero) but the current read tile is of type t_1. As $0+1$ is odd, this tile has an eastern glue with label 1. There are four types of such tiles, that correspond to the four possibilities t_{a+b} where $a, b \in \{0, 1\}$, a represents the parity of the number of tiles of type t_1 read so far, and b represents the type of the currently read seed tile, and the eastern glue of t_{a+b} has label $a + b \bmod 2$. Thus, at the end of the assembly, the second row is filled (until just before the seed tile of type t_b), and the eastern glue label of the north-eastern tile has value 1. The assembly of Figure 3.22 is similar, with the binary string "10001", and so the outcome is an eastern glue label of 0.

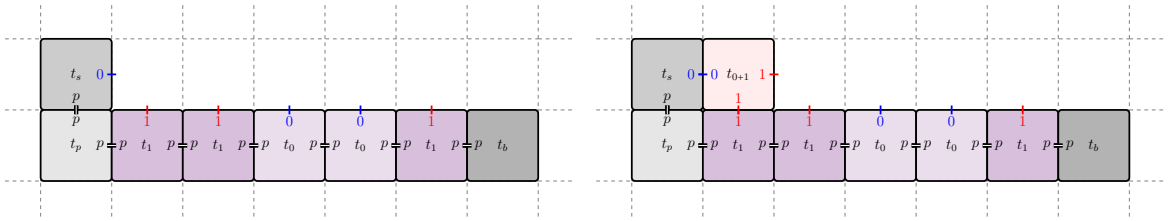


(a) Parity tiles.



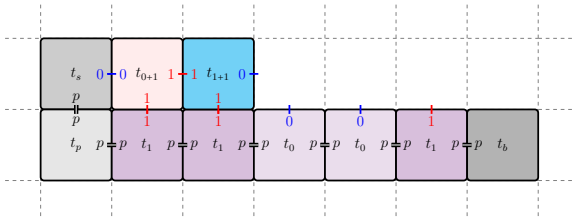
(b) Seed tiles.

Figure 3.20: The tile types from the TAS $\mathcal{S}_p$ of Example 3.34.

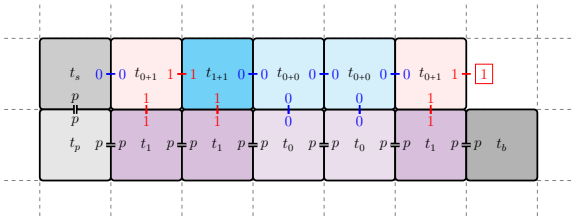


(a) A seed assembly with three tiles of type t_1.

(b) First step of the assembly.

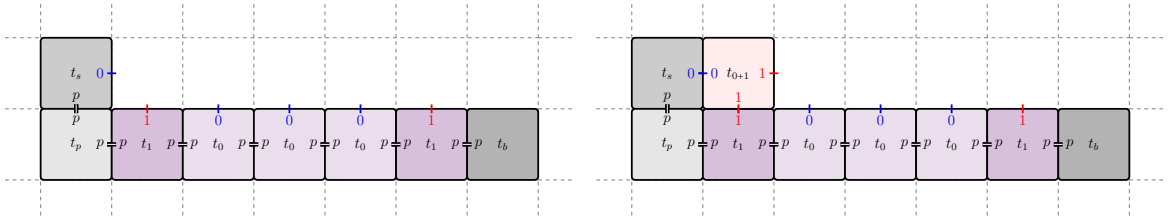


(c) Second step of the assembly.



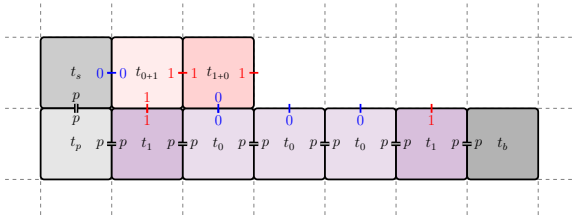
(d) Final step of the assembly.

Figure 3.21: Stages of a terminal assembly for the parity aTAM from Example 3.34, for input 11001. Unlike the general case, here the order of the assembly is fully deterministic.

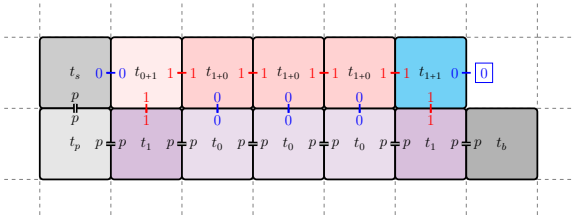


(a) A seed assembly with two tiles of types t_1.

(b) First step of the assembly.



(c) Second step of the assembly.



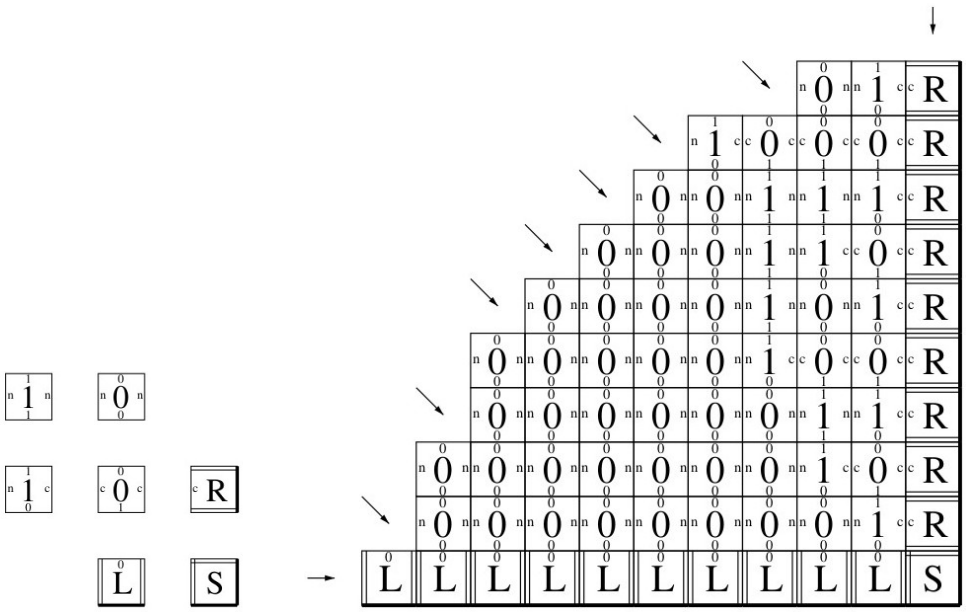
(d) Final step of the assembly.

Figure 3.22: Stages of a terminal assembly for the parity aTAM from Example 3.34, for input 10001. Unlike the general case, here the order of the assembly is fully deterministic.

Our next example is a more complicated binary counter, taken from [65].

Example 3.35 (Binary counter TAS [65])**.** *Let* $\mathcal{S}_b = (\Sigma, T, \alpha_\sigma, str, \tau)$ *be the aTAM TAS:*

- $\Sigma = \{0, 1, c, n, \|, \|\}$ *is the alphabet;*
- $T = \{S, R, L, t_{0+c}, t_{1+n}, t_{0+n}, t_{1+c}\}$ *(see Figure 3.23 for the tile types);*
- *the seed assembly* α_σ *is composed of a tile of type* S *at* $(0,0)$*, a horizontal sequence of tiles, each of type* L *starting at* $(-1,0)$*; and a vertical sequence of tiles, each of type* R *starting at* $(0,1)$*.*
- $str(0) = str(1) = str(c) = str(n) = 1$*,* $str(\|) = 0$ *and* $str(\|) = 2$*;*
- *the temperature* $\tau = 2$*.*



(a) Tile types of $\mathcal{S}_b$. (b) Example assembly for the TAS $\mathcal{S}_b$.

Figure 3.23: The binary counter TAS $\mathcal{S}_b$ from Example 3.35, with the seven tile types on the left, and an example assembly on the right. Image taken from [65].

The tile types of the TAS $\mathcal{S}_b$ from Example 3.35 are depicted in Figure 3.23. They include S, R, L, and four tile types that are labeled by their associated values: 1 for t_{0+c} and t_{1+n}, and 0 for t_{0+n} and t_{1+c}.

The TAS $\mathcal{S}_b$ implements an unbounded binary counter, starting from the binary representation of the number "1" in the second row, and increasing by 1 in each row. The seed is made of the tile type S, a horizontal sequence of tiles of type L and a vertical sequence of tiles of type R that are bounded to each other by strength 2 label ||. Since the temperature is 2, the only position where a tile can first be placed is at the north-west side of the tile of type S, and it must be a tile of type t_{0+c}. Then, tiles of type t_{0+c} and t_{1+c} will fill the vertical column at its north side; and tiles of type t_{0+n} form a horizontal column of tiles at its west side. More precisely, the first tile to bind in each row is the easternmost. Once a tile of type t_{0+c} appears, tiles of type t_{0+n} continue the row to the west. Each tile is bound by its eastern and southern neighbors, i.e. in order to have a tile of type t_* with the value of its northern label as $a + b$, there is a tile at its south with northern label a, and there is a tile at its east with western label b. The carry over of the sum appears in the western label of the tile of type t_*. Moreover, at each step, the tiles of type R cause the increment of the row below, thanks to the label c. See an example of an assembly in Figure 3.23.

In the PhD thesis of Evans [32], a binary counter was implemented using DNA tiles. See Figure 3.24 for an illustration.

3.3.4 Another example: simulating a Turing machine

Next, we show how an aTAM TAS can simulate a Turing machine to perform some computations. We recall the definition of a Turing machine, imagined in 1937 by Alan Turing [70].

Definition 3.36 (Turing machine)**.** *A (deterministic)* Turing machine $M = (Q, \Gamma, q_0, \delta, F)$

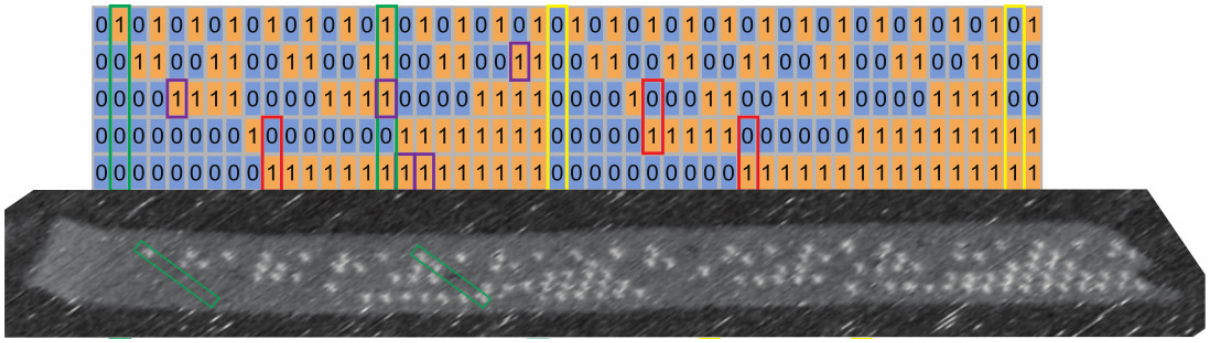


Figure 3.24: The physical implementation of a binary counter (going from left to right), taken from [32]. The red, yellow, green and purple squares represent bits where an error happened.

consists of an alphabet Γ*, a set* Q *of states, where* $q_0 \in Q$ *is the* start state *and* $F \subseteq Q$ *is a set of* accepting states*, and a* transition function $\delta : Q \times (\Gamma \cup \{\sharp\}) \to Q \times (\Gamma \cup \{\sharp\}) \times \{\leftarrow, \rightarrow, \bullet\}$*, where* $\sharp$ *is a* blank symbol *with* $\sharp \notin \Gamma$*.*

M has an infinite discrete tape *(represented by* $\mathbb{Z}$*) and a* head *positioned on the tape, that can read and write.*

Initially, the head is positioned at the coordinate 0 *of the tape, and a finite input string from* Γ^k *is written at coordinates* $0, \ldots, k-1$ *of the tape. All the other coordinates of the tape contain the blank symbol* $\sharp$*. The current state is the start state* q_0*.*

The computation is done following the transition function δ*: if the machine is in state* q *and the symbol* $s \in \Gamma \cup \{\sharp\}$ *is at the head's coordinate, and* $\delta(q, s) = (q', s', m)$*, the machine goes into state* q'*, writes symbol* s' *on the current coordinate of the tape, and moves to the left (if* $m = \leftarrow$*), right (if* $m = \rightarrow$*) or does not move (if* $m = \bullet$*). If* $q' \in F$*, the Turing machine* accepts *the input and the computation stops.*

The concept of a Turing machine is a fundamental model of any computing process. Indeed, Turing showed that there exists a *universal Turing machine* M_U that can take as an input, the description of another Turing machine M and its input I, such that M_U simulates the computation of M on input I, and accepts the input (M, I) if and only if M accepts I.

We next give an example of a simple Turing machine, that, given an input string over the binary alphabet $\{0, 1\}$, erases all the 0's at the end of the input.

Example 3.37 (Turing machine that deletes all final 0's)**.** *Let* $M = (Q, \Gamma, E, \delta, \{F\})$ *be a Turing machine that removes all the final zeros of a binary string, defined as follows.*

- $Q = \{H, E, F\}$ *is the set of states,*
- $\Gamma = \{0, 1\}$ *is the alphabet and* $\sharp$ *is the blank symbol,*
- $E \in Q$ *is the start state,*
- $F \in Q$ *is the final state, and*
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow, \bullet\}$ *is the transition function with:*
  $\delta(H, 1) = (H, 1, \rightarrow)$
  $\delta(H, 0) = (H, 0, \rightarrow)$
  $\delta(H, \sharp) = (E, \sharp, \leftarrow)$
  $\delta(E, 1) = (F, 1, \bullet)$
  $\delta(E, 0) = (E, \sharp, \leftarrow)$
  $\delta(E, \sharp) = (F, \sharp, \bullet)$

The Turing machine M works by first reading the input string, moving the head to the right until it reaches a blank symbol. During that stage it stays in state H. When it reads a blank symbol, it goes to state E and starts going back, replacing the 0's by a blank symbol. When it reaches a 1, it goes to state F and stops. The *space-time diagram* of the machine M for the input 101000 is presented in Table 3.25. On this diagram, the bottom row represents the initial tape, and each further row represents the tape during the next computation of M. The orange cells represent the positions of the head on the tape.

Final step	1	0	1	#	#	#	#
Step 10	1	0	1	#	#	#	#
Step 9	1	0	1	0	#	#	#
Step 8	1	0	1	0	0	#	#
Step 7	1	0	1	0	0	0	#
Step 6	1	0	1	0	0	0	#
Step 5	1	0	1	0	0	0	#
Step 4	1	0	1	0	0	0	#
Step 3	1	0	1	0	0	0	#
Step 2	1	0	1	0	0	0	#
Step1	1	0	1	0	0	0	#
Sart step	1	0	1	0	0	0	#

Table 3.25: The space-time diagram of Turing machine M for input string 101000.

Next, we show how to simulate the computations of M using an aTAM TAS. To do so, the assembly will represent the space-time diagram of M. We will represent the input string by a seed assembly. The assembly is terminal if and only if the Turing machine reaches a final state. The simulation of Turing Machine M of Example 3.25 via an aTAM TAS is shown in Figure 3.26 and the details of the process are presented in Example 3.38.

Example 3.38 (TAS simulating the Turing machine from Example 3.37)**.** *The TAS* $\mathcal{S}_M = (\Sigma, T, \alpha_\sigma, str, \tau)$ *is defined as follows.*

- $\Sigma = \{0, 1, R, L, H0, H1, E, E0, E1, Final\}$,
- *the set* $T = \{t_{L,0}, t_{L,1}, t_{R,0}, t_{R,1}, t_{\boldsymbol{\nearrow},0}, t_{\boldsymbol{\nearrow},1}, t_{\boldsymbol{\curvearrowright},0}, t_{\boldsymbol{\curvearrowright},1}, t_{\boldsymbol{\curvearrowleft},\#}, t_{\boldsymbol{\swarrow},0}, t_{\boldsymbol{\swarrow},1}, t_F, t_{\#}, t_{\boldsymbol{\nearrow},\#}, t_{\boldsymbol{\curvearrowleft},\#}\}$ *of tile types is shown in Figure 3.27;*
- *the seed assembly* α_σ *consists of a row of tiles* $t_{\boldsymbol{\nearrow},1}, t_{L,0}, t_{L,1}, t_{L,0}, t_{L,0}, t_{L,0}$ *and* $t_{\#}$ *(see Figure 3.28(a) for an example);*
- *the strength function is* $str(0) = str(1) = str(R) = str(L) = str(E) = str(\#) = 1$ *and* $str(H0) = str(H1) = str(E0) = str(E1) = 2$*, and* $str(Final) = 0$*; and*
- *the temperature is* $\tau = 2$.

See Figures 3.28, 3.29 and 3.30 for an example of an assembly, where the input binary string is 101000. The assembly starts with the seed assembly α_σ presented in Figure 3.28(a). From the northern strength 2 labels of the tile of type $t_{\boldsymbol{\nearrow},1}$, the head's more to the right is represented by the tile of type $t_{\boldsymbol{\curvearrowright},1}$ (Figure 3.28(b)). The arrows $\curvearrowleft$, $\curvearrowright$, $\boldsymbol{\nearrow}$ and $\boldsymbol{\swarrow}$ on the tile types represent the simulation of the movement of the head of M (left/right),

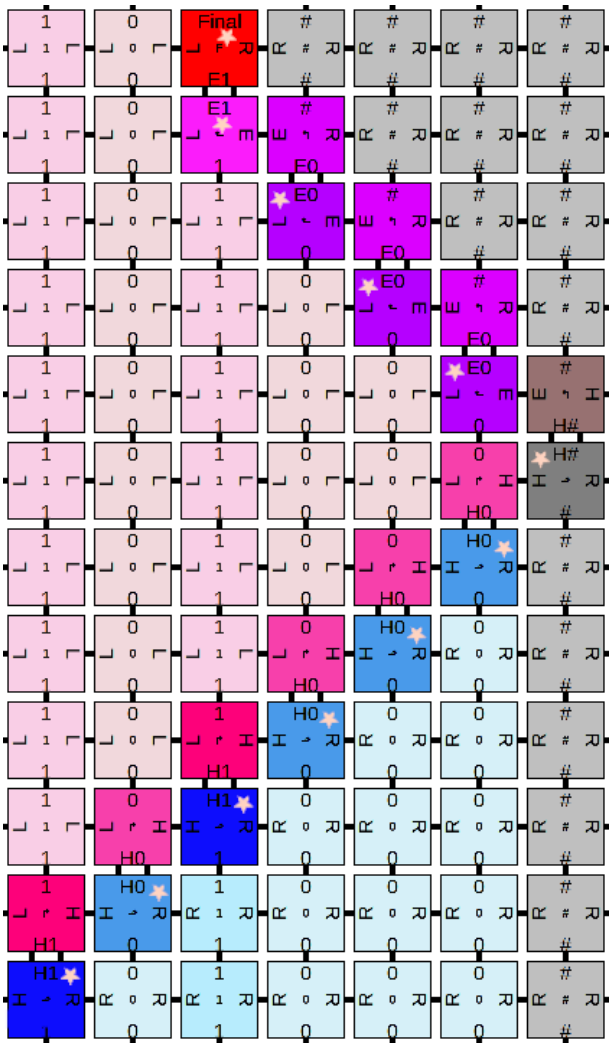


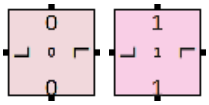
Figure 3.26: The simulation of the computation of Turing Machine M of Example 3.25 by its space-time diagram, via the aTAM TAS $\mathcal{S}_M$, with input string 101000. The tiles representing the position of the head are marked by a star.

jointly with the direction of the assembly (top/down). In the second step, the head is represented by the tile of type $t_{\nearrow,0}$ and the row is filled by tiles of types $t_{R,0}$, $t_{R,1}$ and $t_{\sharp}$. The following rows continue like this: the head tiles of type $t_{\nearrow,0}$ or $t_{\nearrow,1}$ are replaced respectively by tiles of type $t_{\rightarrow,0}$ or $t_{\rightarrow,1}$, and each row fills with tiles of type $t_{L,0}$ and $t_{L,1}$ at the left of the head, and of type $t_{R,0}$, $t_{R,1}$ and $t_{\sharp}$ at the right of the head, such that the rightmost tile is of type $t_{\sharp}$.

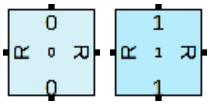
Once the head arrives to a tile of type $t_{\sharp}$, in the next step a tile of type $t_{\nearrow,\sharp}$ is the rightmost tile and represents the head. Then, via the rightmost tile of type $t_{\nwarrow,\sharp}$ the left movement starts. In this step (step 7), the rightmost 0 is erased. In the following, each row opens by a tile of type $t_{\nwarrow,\sharp}$, the tile of type $t_{\leftarrow,E0}$ erases the zero, and the rest of the row is filled by tiles of types $t_{R,0}$, $t_{R,1}$ and $t_{\sharp}$. When there is no tile of type $t_{R,0}$ left to erase, a tile of type $t_{\leftarrow,E1}$ leads the assembly to the appearance of the tile type t_F. Since this tile has no strength 2 label, no new row can be created, and after the topmost row is filled, the assembly is terminal.

Recall that by Remark 3.32, tile self-assemblies are not necessarily deterministic. In this case, although M is a deterministic Turing machine, the assemblies of $\mathcal{S}_M$ are indeed

not deterministic. For example, in Figure 3.29(c), a tile of the top row is placed before the previous row is filled (it could have happened in reverse order: first the filling of the row, then the new row is started). However, the simulation is correct.

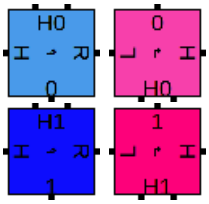


(a) Tiles types $t_{L,0}$ and $t_{L,1}$ at the left of the head.

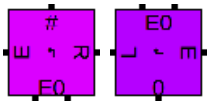


(b) Tiles types $t_{R,0}$ and $t_{R,1}$ at the right of the head.

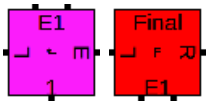
(c) Blank tile types $t_{\sharp}, t_{\rightarrow,\sharp}$ and $t_{\leftarrow,\sharp}$.



(d) Transition tile types $t_{\rightarrow,0}, t_{\rightarrow,1}, t_{\nearrow,0}$ and $t_{\nearrow,1}$ on the head, when the movement is to the right $\rightarrow$.

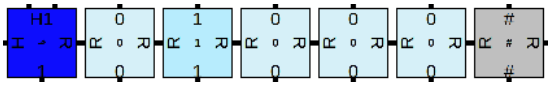


(e) Transition tile types $t_{\leftarrow,\sharp}$ and $t_{\leftarrow,E0}$ on the head, when the movement is to the left $\leftarrow$.

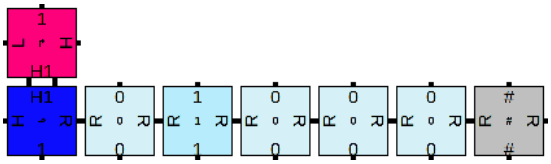


(f) Tile types $t_{\leftarrow,E1}$ and t_F on the head, when there is no movement $\bullet$.

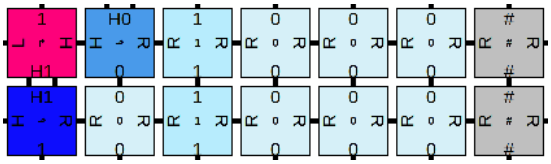
Figure 3.27: The tile types of the TAS $\mathcal{S}_M$ simulating Turing machine M.



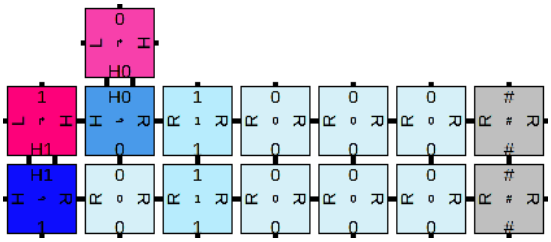
(a) The seed assembly, that represents the input string 101000.



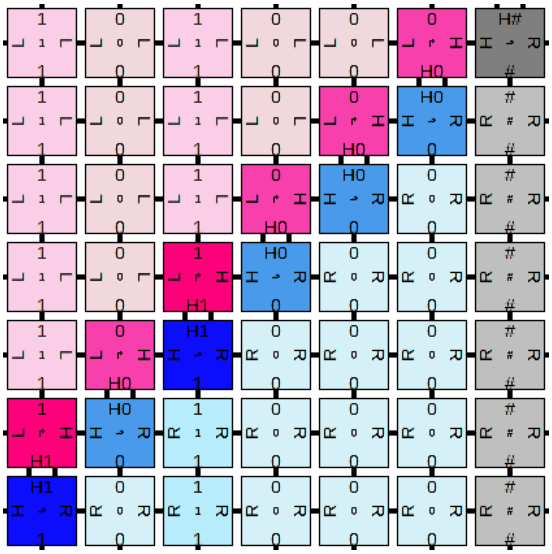
(b) Step 1 of the assembly.



(c) Stage of the assembly that represents the two first steps of the computation.

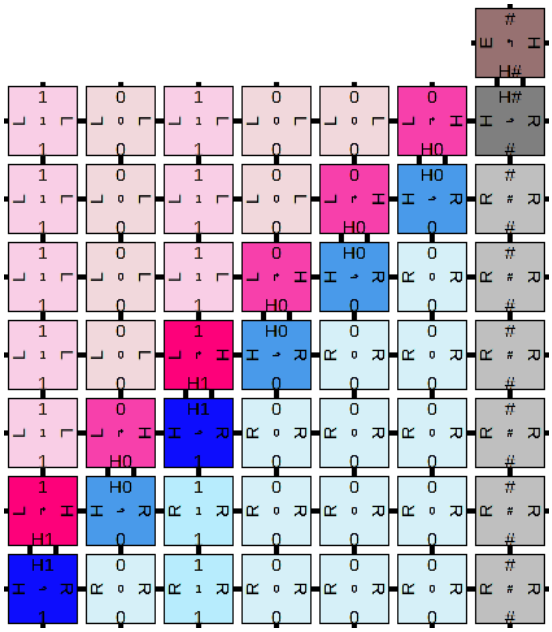


(d) Stage of the assembly that starts the next computation step of M.

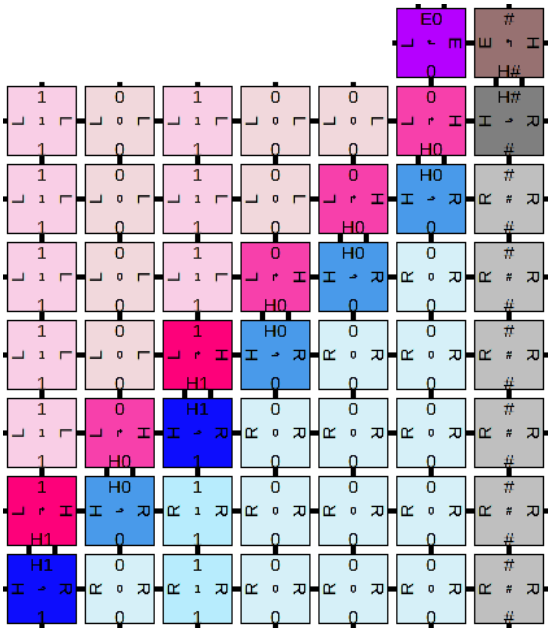


(e) Stage of the assembly representing the computation step of M where the head reached the end of the input.

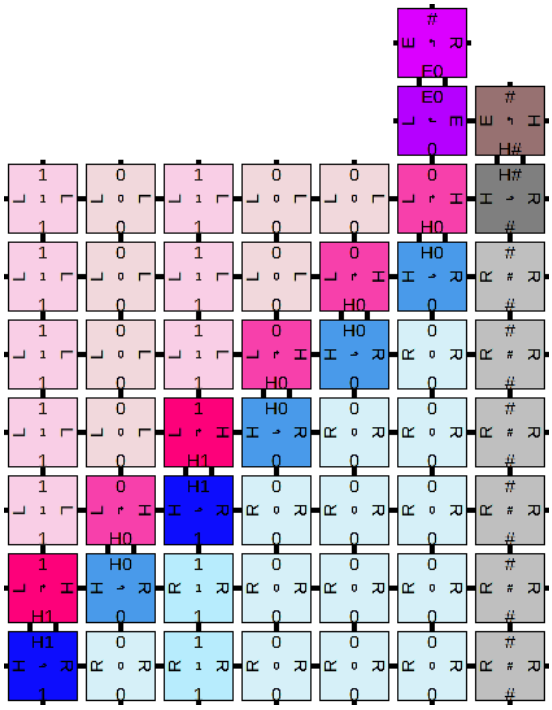
Figure 3.28: First stage of the assembly for the TAS $\mathcal{S}_M$, simulating the Turing machine M for the input string 101000. This stage simulates the computation of M from the start to the point where it finishes to read the input string.



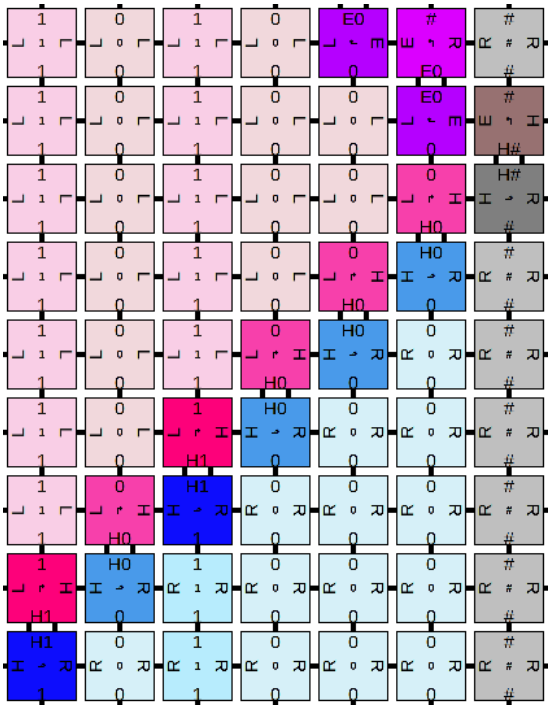
(a) Step of the assembly representing the computation step of M where the head starts going to the left.



(b) Next step of the assembly.

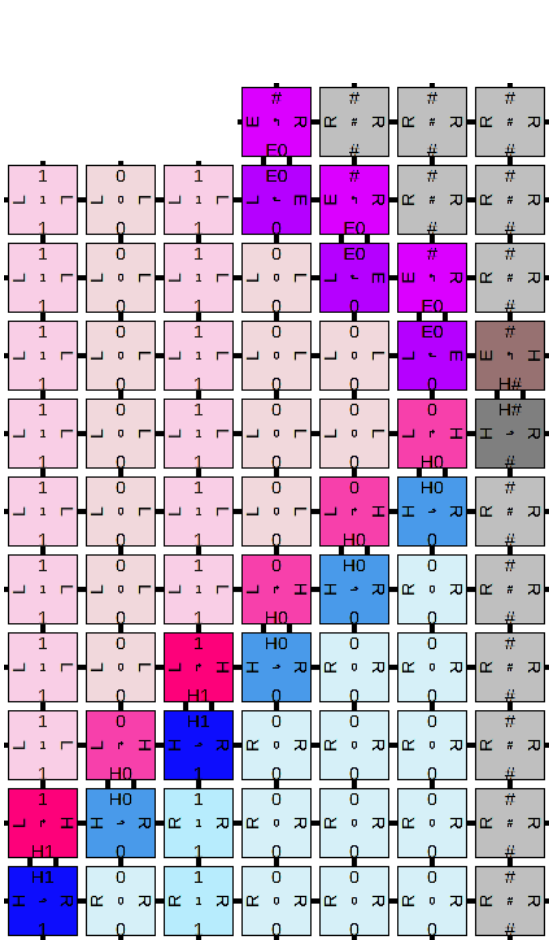


(c) Next step of the asssembly.

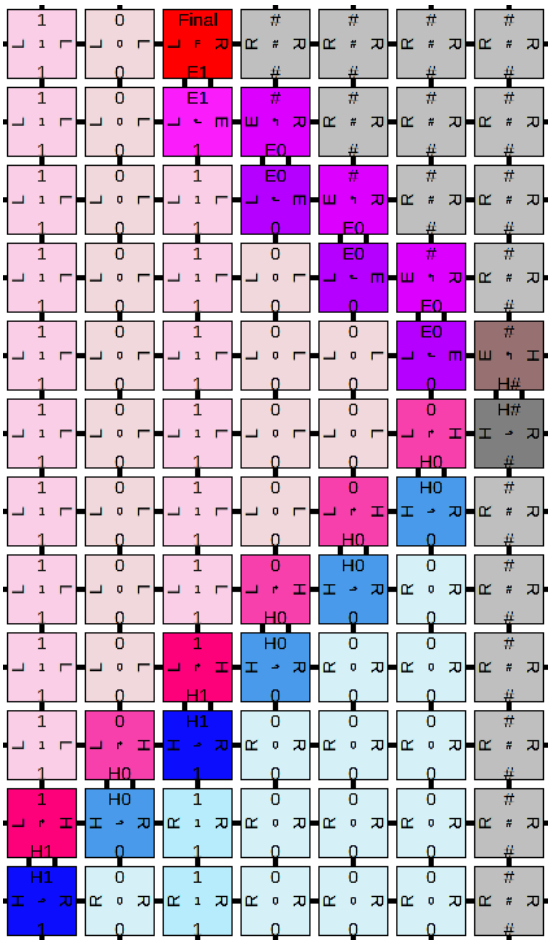


(d) Stage of the assembly after two moves of the head to the left.

Figure 3.29: Second stage of the assembly for the TAS $\mathcal{S}_M$, simulating the Turing machine M for the input string 101000. This stage simulates the computation of M from the point where it finishes to read the input string and erases all the 0's at the end of the input string. (Part 1 of 2)



(a) Stage of the assembly representing the last time a 0 symbol is erased from the tape.



(b) The terminal assembly, when all the final O's have been erased. It is shown by the presence of the red tile at the top of the assembly.

Figure 3.30: Second stage of the assembly for the TAS $\mathcal{S}_M$, simulating the Turing machine M for the input string 101000. This stage simulates the computation of M from the point where it finishes to read the input string and erases all the 0's at the end of the input string. (Part 2 of 2)

3.3.5 Computing using the aTAM

We have seen in Example 3.34 that a simple aTAM TAS can compute the parity of the number of some special tiles in a seed assembly. Moreover, in Example 3.38, we have seen how a TAS can simulate a simple Turing machine. What is the maximum computational power of tile self-assembly? Winfree showed in 1998 that one can perform arbitrary computations using self-assembly, by the following theorem.

Theorem 3.39 (Winfree [75, 76])**.** *There is an aTAM TAS with temperature 2 that can simulate any Turing machine.*

In the language of computational complexity theory, the above theorem says that tile self-assembly in the aTAM is *Turing-universal*: informally speaking, any computation that can be performed by a computer, can be performed by the TAS of this theorem. To do so, the input of the problem is encoded in the seed, and the result of the computation can be read in the terminal assembly of the TAS.

Theorem 3.39 is proved by simulating a class of Turing-universal *cellular automata*, which form a model of computation that also takes place in the d-dimensional lattice (for d-dimensional cellular automata). Each position of $\mathbb{Z}^d$ is called a *cell* and a cell can be in a finite number of states. The model works step by step, and in each step the cells update their state as a function of the states of neighbouring cells. This function is described by a set of *transition rules*. The rules are applied in parallel. It is known that some cellular automata can simulate any Turing machine, and this is true even for specific 1-dimensional cellular automata (as proved by Lindgren and Nordahl in 1990 [47]). The way that a 1-dimensional cellular automaton C can be simulated by a tile self-assembly system, is to associate a type of tile for each transition rule of C, and the southern and northern glues of the tiles represent the sate of the corresponding cell of C at some time steps i (south) and $i+1$ (north). The first row of the assembly will represent the initial states of the cells of C, and at each step, a new row is added to represent the next states of cells of C. Thus, similarly as for Example 3.38, the assembly represents the *space-time diagram* of the cellular automaton C. The terminal assembly represents the final states of cells of C. Thus, C is simulated by such an aTAM TAS, and so the aTAM is Turing-universal.

One can also directly simulate any Turing machine, similarly as the simulation of a specific Turing machine from Example 3.38 and the simulation of a 1-dimensional cellular automaton described above, but this is more complicated. We refer to the 2011 paper of Lathrop et al. [46] for details.

Winfree, Xu and Seeman experimentally verified in 1996 [77] that the crucial steps for the construction can indeed be done using real DNA tiles. Moreover, it was shown experimentally that practical DNA-based tile self-assembly systems can solve many tasks such as *"copying, sorting, recognizing palindromes and multiples of 3, random walking, obtaining an unbiased choice from a biased random source, electing a leader, simulating cellular automata, generating deterministic and randomized patterns, and counting to 63"* (Woods et al., 2019 [78]).

The type of theoretical simulation results as described above has been refined even more, for example, Doty et al. showed in 2012 [29] that there exists a single tile assembly system that simulates all other systems at temperature 2; this property is called *intrinsic universality*.

On the other hand, it was proved by Meunier and Woods [54] that at temperature 1, the aTAM is not intrinsically universal. Even more, Meunier, Regnault and Woods [52]

showed that precise control of the assemblies at temperature 1 is impossible, and thus the type of results described in the previous paragraphs cannot be obtained at temperature 1. Meunier and Regnault also showed that aTAM tile assembly systems at temperature 1 with only one terminal assembly are decidable, and thus, cannot simulate all Turing machines [53].

3.3.6 Creating specific shapes

A large portion of the work in self-assembly is to design tile self-assembly systems that result in specific shapes, in the sense that their terminal assembly forms the desired shape.

Perhaps the most natural shape that can be assembled is an $n \times n$ square. This simple shape has been used as a reference shape in many works, to compare the different models and solutions. See Figure 3.31 for a TAS with $O(\log n)$ tile types that assembles an $n \times n$ square, designed in 2000 by Rothemund and Winfree [65].

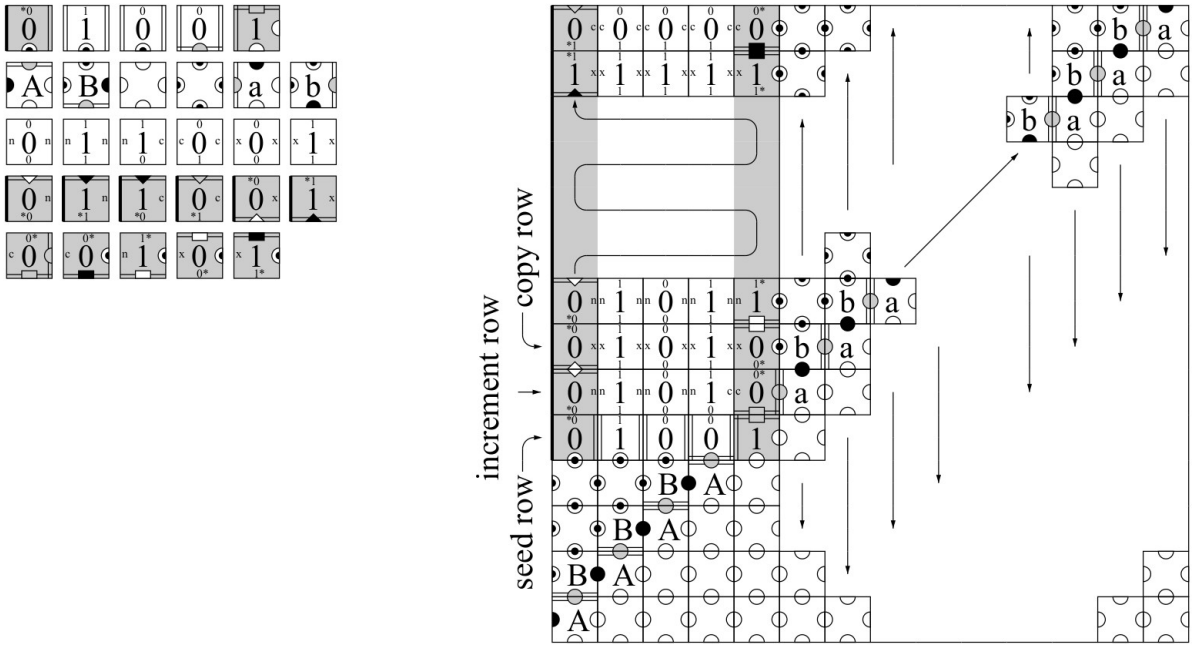


Figure 3.31: A TAS for assembling an $n \times n$ square, with an illustration of the assembly. Image taken from [65].

Another fundamental shape for which a TAS was designed in 1998 by Winfree [75] is the Sierpinski triangle. See Figure 3.32 for an illustration. Here, the growth of the assembly is controlled by the fact that new tiles can only attach if two previous tiles are present (one on the west and one on the south), similarly as the assemblies from the TAS of Example 3.26, where south and west are replaced by north and east. Hence, there is no need to enforce synchronization in this assembly.

3.3.7 Complexities of assemblies

As we have seen in Section 3.3.5, any computation that can be performed using a classical computer, can be done by a carefully designed TAS. Thus, most shapes can be formed as well, as long as they can be computed by a computer. However, a TAS that builds the shape could be very complex, and the Turing-universality result by Winfree is based on

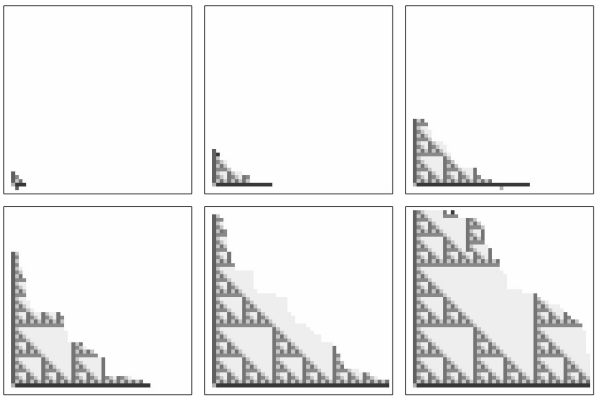


Figure 3.32: The assembly of a Sierpinski triangle, from [75].

the simulation of a cellular automation that itself simulates a Turing machine, hence the resulting TAS is inherently very complex. One can measure the complexity along different measures, as proposed by Rothemund and Winfree in 2000 [65] as follows.

Definition 3.40 (Tile complexity)**.** *The* tile complexity *of a shape S is the smallest number of tile types in an aTAM TAS whose unique terminal assembly is S.*

For a family F of shapes, the tile complexity *of F is the smallest number of tile types in an aTAM TAS whose terminal assemblies are exactly the shapes in F.*

For example, the tile complexity for the family of all squares is 5, and the one for the family of all rectangles is 6 [12, Chapter 4]. It was shown in 2001 by Adleman et al. [3] that the tile complexity of a single (fixed) square of dimension $n \times n$ is in $\Theta\left(\frac{\log n}{\log \log n}\right)$.

However, for general shapes, determining the optimal tile complexity of a given shape is an NP-hard problem, as shown by Adleman et al. in 2002 [4], which means it is computationally difficult to determine the optimal tile complexity of a shape.

Besides the number of tiles, one can also count the optimal number of steps of an assembly, for a given shape. Here, it is assumed that two tiles that do not interact can attach at the same time, and the maximum time is thus the largest sequence of tiles such that two consecutive tiles in the sequence interact.

Definition 3.41 (Time complexity)**.** *The* time complexity *of a shape S is the smallest number of steps needed by a TAS whose unique terminal assembly is S.*

For a family F of shapes, the time complexity *of F is the smallest number of steps needed by a TAS whose terminal assemblies are exactly the shapes in F.*

For example, in 2008, Becker et al. designed in [16] a TAS with provably optimal time complexity, for assembling the family of all squares. Indeed, they designed an aTAM TAS with 24 tile types (with rotation allowed) that can produce any $n \times n$ square in $2n$ time steps, which is optimal. For a single (fixed) $n \times n$ square, the time complexity is also in $\Theta(n)$ [3].

A larger class of shapes is considered by Becker et al. in 2006 [15] (rectangles, squares and diamonds).

3.4 Tile self-assembly on 2D surfaces other than the plane

In this thesis, we are interested in performing self-assembly on existing 3D surfaces, such as polycubes. We are not aware of any previous work on this topic. However, there have been some recent works on DNA tile self-assembly where also, the self-assembly is done on an existing surface. We now present these works.

3.4.1 Tile self-assembly on mazes

One recent work of interest, where the self-assembly is performed on a complicated surface, is the one of self-assembly in *mazes*.

In 2021, Cook et al. [27] defined a new self-assembly model, where the tile placement is done along the walls of a certain maze, as depicted in Figure 3.33. This model is called the *Maze-Walking Tile Assembly Model* (Maze-Walking TAM for short). It is similar to the aTAM, but the TAS contains a fixed maze, and they allow multiple disconnected seeds (one for each entry of the maze). Tiles attach to the maze if the glues on two sides of the tile match the glues of the maze walls. In this process, the time complexity of the assembly partially depends on the environment.

Intuitively speaking, exploring a maze is seen in [27] as a computational process: the maze represents the constraints and the problem can be solved if there is a path from the start to the end.

The authors of [27] show that any *Boolean circuit* can be simulated by a TAS in the Maze-Walking TAM using only four tile types. This shows that this model enables to perform arbitrary computations by putting together several mazes.

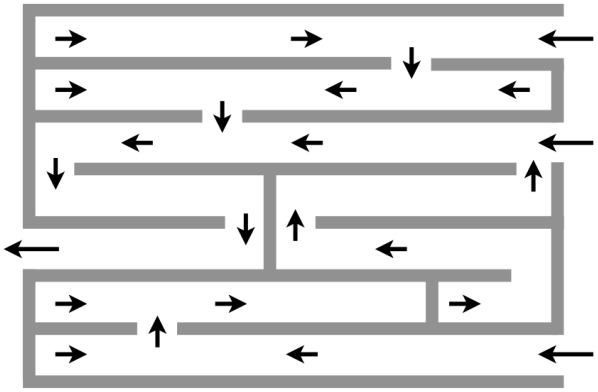
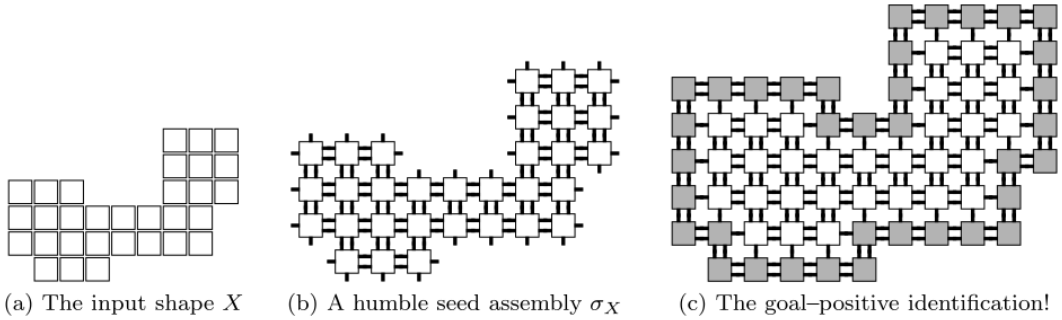


Figure 3.33: A maze on which self-assembly is performed, from [27].

3.4.2 Shape identification

In 2010, Patitz and Summers studied the problem of *identifying* a given 2D shape (in the form of a seed assembly) [60]. They work in the variant of the aTAM called *two-handed aTAM*. In their work, a TAS correctly identifies the target shape if a special type of tiles appears along the entire border of the shape when the seed assembly matches the target shape, and the border does never form when the seed assembly does not match the target shape. See Figure 3.34 for an illustration (taken from [60]).



(a) The input shape X (b) A humble seed assembly σ_X (c) The goal–positive identification!

Figure 3.34: Illustration of the shape identification framework studied in [60], in the case where the input shape matches the target shape and is correctly identified.

The authors of [60] design a TAS in this model that can correctly identify an $n \times n$ square using $O\left(\frac{\log n}{\log \log n}\right)$ tile types. They also design a TAS with $O(1)$ tile types that can identify whether the shape is a square (of any size). They also design TASes for a more general class of certain hole-free shapes.

3.4.3 Shape replication

Another interesting type of work where the self-assembly is made along an existing arbitrary surface (called a shape) is the topic of *shape replication*.

In 2010, Abel et al. [1] used a variant of the aTAM called *enzyme self-assembly model* to implement shape replication. In this model, enzymes can be used to *destroy* tiles of a given tile type, that have been placed previously. The goal is that a preexisting shape or pattern is given, and the tiles react to the shape in order to reproduce it. Here, the assembly takes place on the 1-dimensional border of a 2D pattern. The main challenge is that the system must react to the shape of the space around, rather than to an external input it can read as it wants. See Figure 3.35 for an illustration of the shape replication presented in [1].

In [1], it is shown that one can replicate a given shape in this model using a tile system with a constant number of tile types. They consider both the problem of replicating the shape indefinitely, or replicating it exactly n times, for a given integer n.

In 2017, Chalk et al. [22] showed that the results from [1] can be obtained in a simpler model, where we do not need to destroy some of the tiles using enzymes. However, they use the additional feature of *negative glues*, which enforces that no tile can be attached at the edge of this negative glue.

Similar results have been obtained using a model of *Signal-passing Tile Assembly Model* (STAM), see [7, 8, 44]. The signals in this model, propagate across assemblies to activate or deactivate glues. In [7], Alseth, Hader and Patitz obtain a single TAS in the STAM that can replicate any 2D or 3D shape, and they show that for such a result, the deactivation of glues (and deconstruction of the initial shape) is necessary. They consider the shapes as assemblies. In their model, one needs to deconstruct the input shape in order to replicate it, see Figure 3.36 for an illustration.

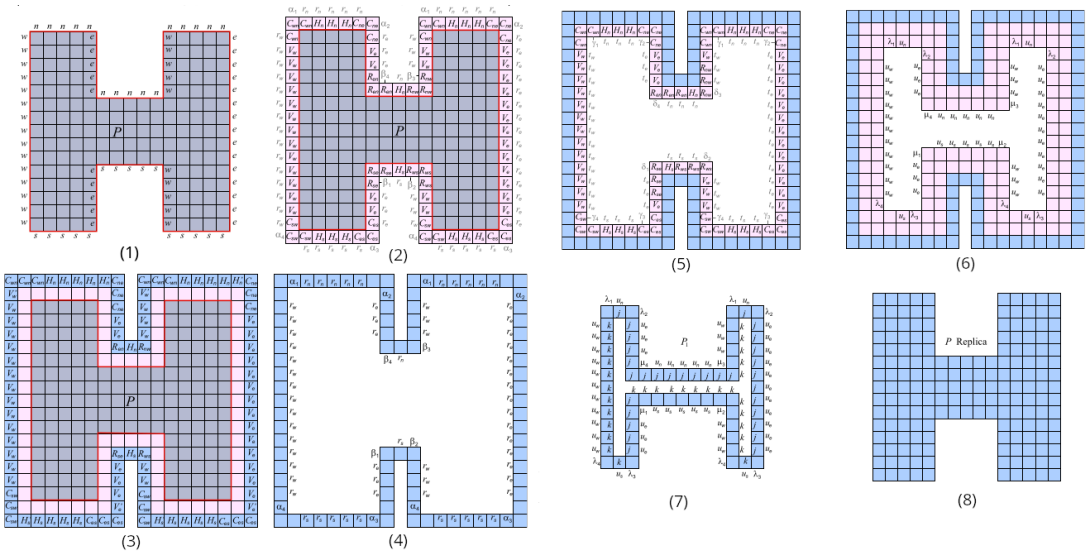


Figure 3.35: Illustration of the shape replication performed in [1]. A first self-assembled RNA-based layer surrounds the shape, followed by a second DNA-based layer. Then, the first layer is destroyed using enzymes, and the second layer becomes free from the initial shape. The second layer can then be filled to obtain a copy of the shape.

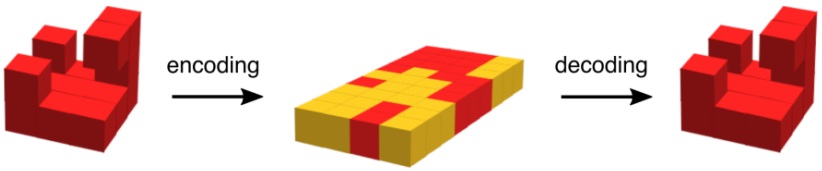


Figure 3.36: The method of shape replication used in [7]. Here, the initial shape is an assembly that may have to be destroyed in the process.

3.4.4 Self-assembled coatings of surfaces

An interesting application of self-assembly is in material sciences, where materials are created using self-assembly of molecules. In one particular class of applications, the assembled materials are *coatings*, that is, layers of molecules around an existing object [21], that give additional properties to the object, such as increased resistance to the environment. These coatings are applied to many different areas, for example self-assembled coatings for the design of drugs [71] and self-assembled coatings on metallic objects [81]. See Figure 3.37 for an illustration from [37].

Although these self-assembly processes are quite different from those studied in this thesis, the setting is similar, since the self-assembly of these coatings takes place on a pre-existing surface.

3.5 Assemblies to construct 3D shapes

We are interested in performing self-assembly on the surface of 3D shapes. We are not aware of such existing works, but many previous works from the literature deal with *constructing* 3D shapes, in various models. Many of them deal with polycubes, which are of special interest to us since we will study them in Chapters 5 and 7. We next present

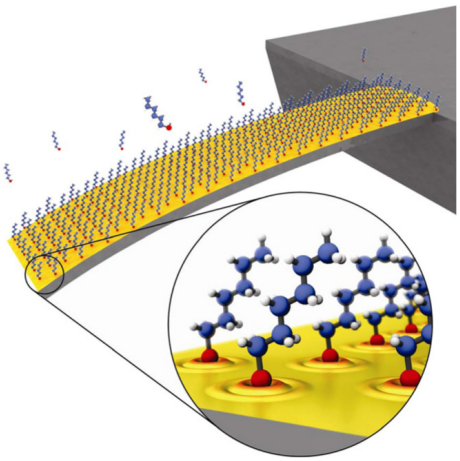


Figure 3.37: Coating a surface of gold using a layer of slef-assembling molecules, from [37].

the works in this area, that we are aware of.

3.5.1 The Flexible Tile Assembly Model (FTAM)

We now describe a model that inspired us to introduce the SFTAM model in Chapter 5. In this model, 3D structures (in particular, polycubes) are assembled using 2D square tiles.

Recently, a new tile self-assembly model called *Flexible Tile Assembly Model* (FTAM) was introduced by Durand-Lose et al. in [30], as an extension of earlier work [16, 43]. Here, we have Wang tiles but they self-assemble (without an input surface) in 3D space (modeled by the lattice $\mathbb{Z}^3$) as they can have, in addition to standard *rigid* bonds, *flexible* bonds that allow tiles to bind at any angle along the tile edges. Because of this flexibility, the assemblies are able to move and fold, which makes the model rather complicated.

In the FTAM, the tile types are as defined in the aTAM (Definition 3.24), and tiles are placed on facets of $\mathbb{Z}^3$. However, it is not enough to know on which facet the tile is placed at, indeed the tiles can rotate and reflect. Thus, in the FTAM, the following definition was given to describe how a tile is placed.

Definition 3.42 (Placement in the FTAM [30])**.** *In the FTAM, a* placement *of a tile* $p = (l, n, o)$ *consists of a location* $l \in \mathbb{Z}^3$*, a normal vector* n *which starts at the center of the tile and points perpendicular to the plane in which the tile lies, and an orientation* o *which is a vector lying in the same plane as the tile which starts at the center of the tile and points to the northern side of the tile.*

The main goal of the work in [30] is to construct polycubes using FTAM tile self-assembly. To do so, the assembly must control the way that the flexible and the rigid bonds interact. FTAM assemblies are either *rigid* or *flexible*. See Figure 3.38 for an illustration of an FTAM assembly sequence, taken from [30].

The first result proved in [30], is that every polycube that satisfies three conditions, can be assembled deterministically using a temperature 2 FTAM TAS. The conditions are the following: (i) the polycube is symmetric, (ii) it contains no vertices such as in Figure 3.9(e) or Figure 3.9(i) and (iii) the edges of the polycube are connected.

Then, they consider computational complexity questions. They show that, given an FTAM TAS, it is undecidable whether this TAS can produce a terminal assembly that

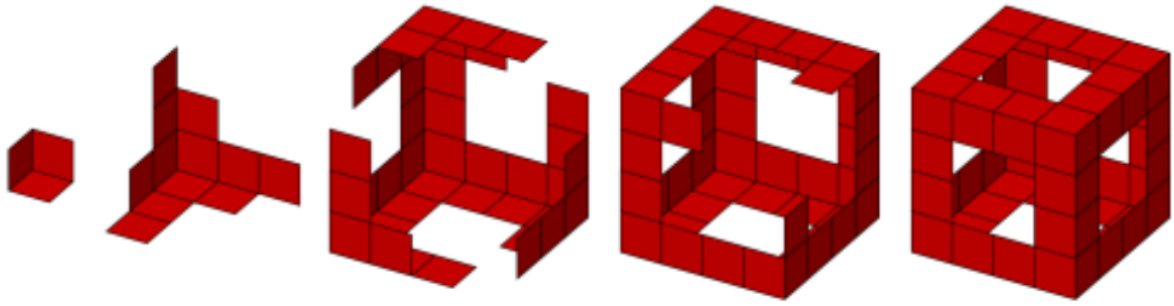


Figure 3.38: An assembly sequence in the FTAM, from [30].

is rigid. They prove this by simulating a Turing machine in the FTAM, and the Turing machine halts if and only if the assembly is rigid.

Moreover, they show that given an FTAM TAS and an assembly α, deciding whether α is rigid is a coNP-complete problem. They prove this by reducing from the complement of the NP-complete Boolean Satisfiability (SAT) problem.

3.5.2 Self-assembly of polycubes using unit cubes as "3D tiles"

We now review some works where 3D polycubes have been self-assembled using cubic 3D tiles.

In some practical applications, instead of unit square tiles, the building blocks for self-assembly are unit cubes, like in [68], where hydrogel cube assembly was performed experimentally.

On the theoretical side, it was shown that when using 3D unit cube tiles, and unlike in 2D, self-assembly is Turing-universal even at temperature 1, as proved in 2001 by Cook et al. [26] with extensions by Furcy and Summers in 2018 [33]. The model that they use is called *3D abstract Tile Assembly Model* and it is like the traditional aTAM, but with unit cube tiles instead of unit square tiles. Like in the aTAM, they can bind by their faces if the two binding faces are similar.

In 2008, Becker et al. [12, 16] used a similar model, where the basic tiles are 3D unit cubes. In these works, the time complexity of assembling the family of all $n \times n \times n$ cubes is studied, and they designed a TAS in temperature 3 that produces this family in optimal time $3n$. See Figure 3.39 for an illustration of an assembly of a cube in this model.

A similar model is used by Bohlin et al. in 2022 [18, 19], however, there, the model is stochastic, that is, they require the assemblies to be produced with a certain (high enough) probability. The problem studied in these works is how to design such self-assembly systems in order to assemble specific polycubes. More precisely, they provide an "assembly pipeline" that, given a polycube, automatically computes a TAS able to assemble the given polycube with high probability. For this, they formulate the problem in the language of Boolean Satisfiability (SAT) and use dedicated tools to solve the corresponding SAT problems. They perform simulations using self-assembly simulation softwares "oxView" and "oxDNA". See Figure 3.40 for an illustration of an assembly in this model, taken from [18].

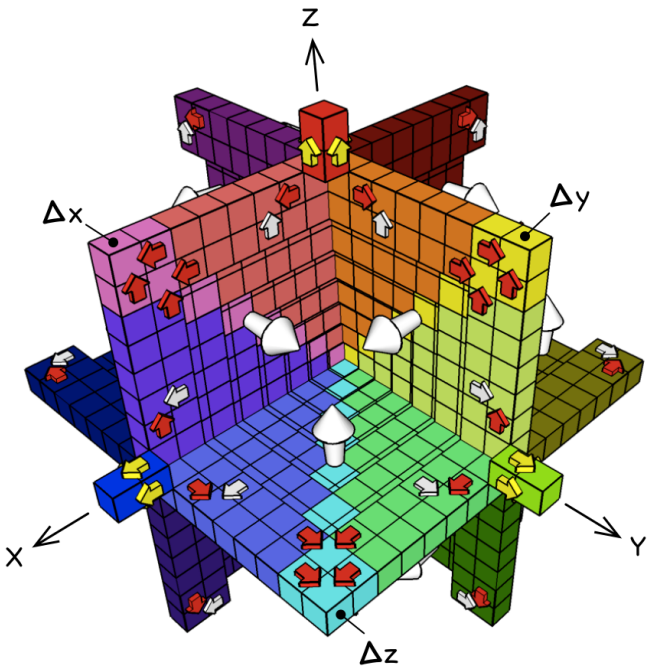


Figure 3.39: The assembly of a cube using cube-shaped "3D tiles", from [12, 16].

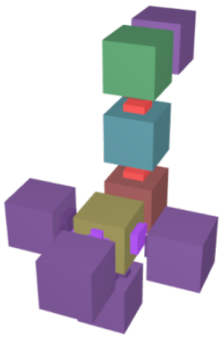
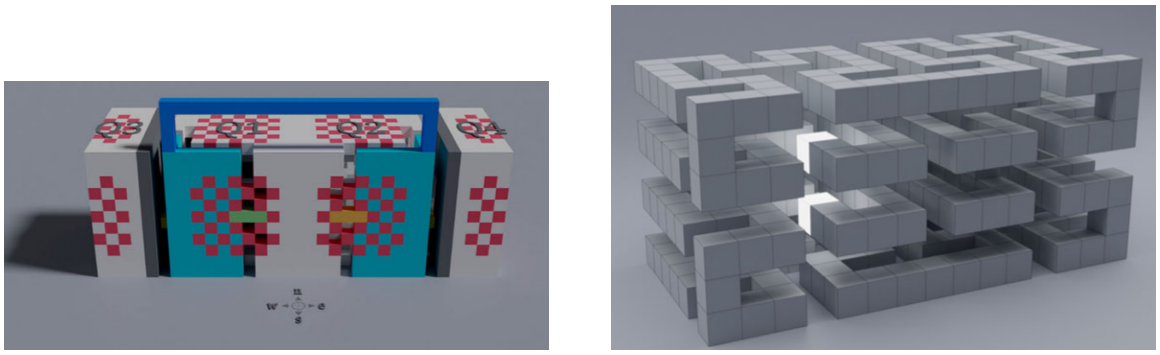


Figure 3.40: A polycube assembly using cube-shaped "3D tiles", from [18].

3.5.3 Particle-based assembly

Another related work (dealing with assembly, but not *self*-assembly) has been used to assemble polycubes. Here, unit cubes called *particles* bind along their faces, but they have no distinguished sides. They are released one by one and guided (in practice, this can be done using electric fields, see Becker et al., 2020 [11]) to their eventual final location, which is adjacent to a particle that is already placed; then, the two particles bind, in the same way as in self-assembly models. In some works, it was studied how specific polycubes can be constructed (or not) using this model. See Figure 3.41 for two polycubes addressed by Keller et al. in 2022 [45].

In [45], the authors study the variant where the path of the particle is an arbitrary connected path. They show that deciding whether a given 3D shape can be constructed, is an NP-complete problem. However, for the shapes that are tree-like, they design a polynomial-time algorithm. In [11], the particle path has to be a straight line. The authors prove that it is NP-hard to decide if a given polycube can be constructed in this model.



(a) A polycube that can be built in the particle model from [45].

(b) A polycube that cannot be built in the particle model from [45].

Figure 3.41: Two polycubes from [45].

3.5.4 Crystal self-assembly

Another type of 3D self-assembly are those creating complex molecules, and in particular, crystals [10, 82, 50]. In these experimental works, the goal is to design self-assembly systems based on basic components resembling tiles, that can grow crystals of various types. One important work is a construction built from basic DNA triangles [48]. See Figure 3.42 for a schematic view of such an assembly, from [82]. A recent perspective article about these crystal assemblies is available in [50].

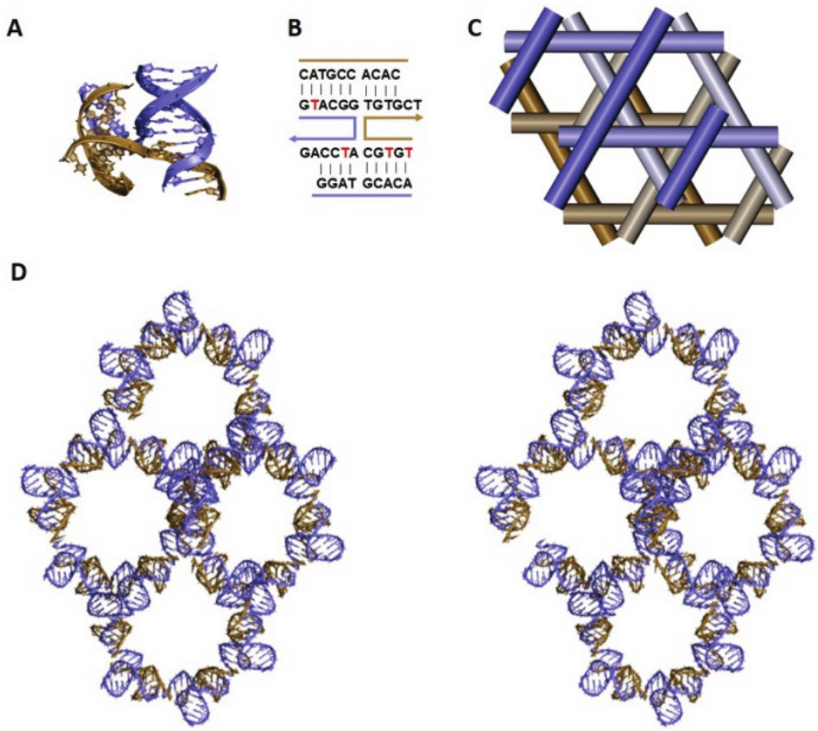


Figure 3.42: Schematic view of the assembly of a DNA crystal lattice, from [82].

3.5.5 Origami-like folding processes

We now describe some related assembly processes which have been used to produce 3D shapes, in particular, polycubes.

Inspired by the Japanese traditional art of *origami*, Rothemund invented in 2006 [64] the concept of *DNA origami*, where a long DNA strand is folded into a given shape and "stapled" together using multiple short strands. He used this model to produce experimentally various kinds of 2D shapes, see Figure 3.43. This model has then been extended to enable the design of 3D objects, see [38] (see Figure 3.44 for an illustration).

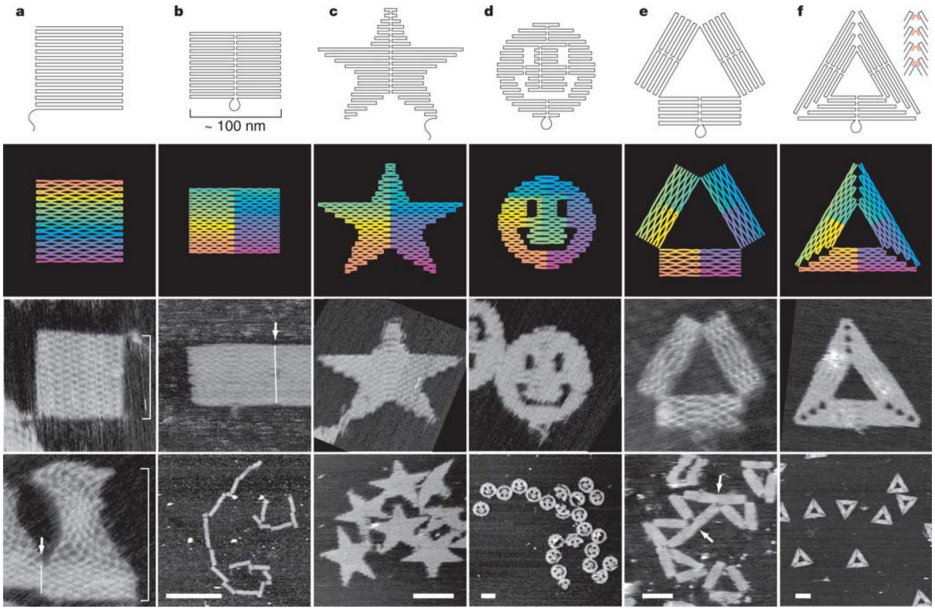


Figure 3.43: Shapes assembled by DNA origami, from [64].

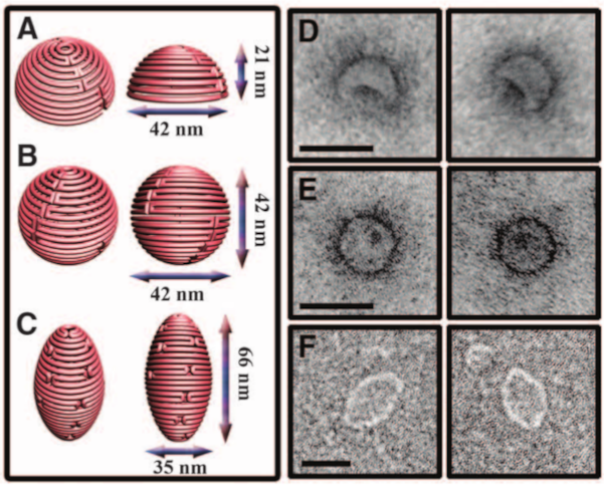


Figure 3.44: 3D objects assembled by DNA origami, from [38].

A related topic is studied in 2010 by Aloupis et al. in [6]. An *unfolding* of a polycube is a connected planar arrangement of unit squares obtained by cutting the polycube along some of its edges. One question studied in [6] is whether one can find a common unfolding for a large class of polycubes (see Figure 3.45 for an example of a common unfolding of two

polycubes). More precisely, the authors prove that the class of all *"tree-like" pentacubes* (polycubes made of five unit cubes, and whose shape is tree-like) do *not* have a common unfolding. However, they show that planar tree-like "non-spiralling" polycubes made of k unit cubes, do have a common unfolding.

More recently, in 2018, Aichholzer et al. [5] studied a generalized version of the problem, where folding is allowed on diagonals also. They introduce several different models with various constraints, with the goal of folding a 2D shape into a given polycube. They characterized the 2D shapes that can be folded into a unit cube under these different models.

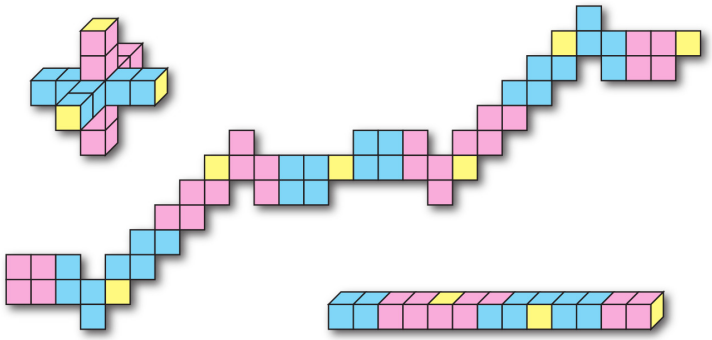


Figure 3.45: A common unfolding of two tree-like polycubes, from [6].

A game called *cubigami* and that inspired work as that from [6] was designed by Knuth and Miller [55], see Figure 3.46.

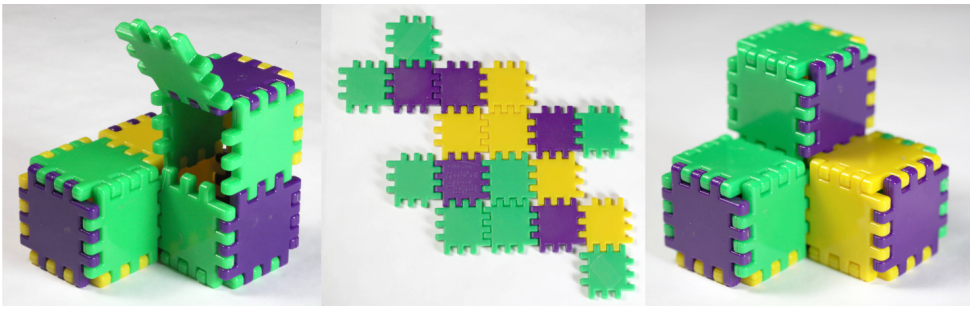


Figure 3.46: The game *cubigami* designed by Miller and Knuth [55].

Another model is called *Oritatami* and was introduced in 2019 by Geary et al. [35]. In this model, similarly as in the aTAM, we are placing elements on a lattice. Here, however, the authors consider a *triangular* lattice. The elements are called *beads* that, like the tiles in the aTAM, can be from a fixed finite set of types. The beads attach as a sequence to form a path-like construction, and the last bead of the sequence is able to fold according to some specified *attraction rules*. See Figure 3.47 for an example of an oritatami configuration that simulates a binary counter, from [35].

The authors of [35] introduced the model and described how to simulate a binary counter in the model, thus showing that one can possibly use this model to compute. However, they also show that designing an Oritatami system that folds into a prescribed set of shapes is an NP-hard problem. Then, in [36], the same authors proved that the Oritatami model is Turing-universal. More recently, Pchelina et al. [61] showed that the

shapes that can be folded in the Oritatami model are as complex as the ones that can be assembled in the aTAM.

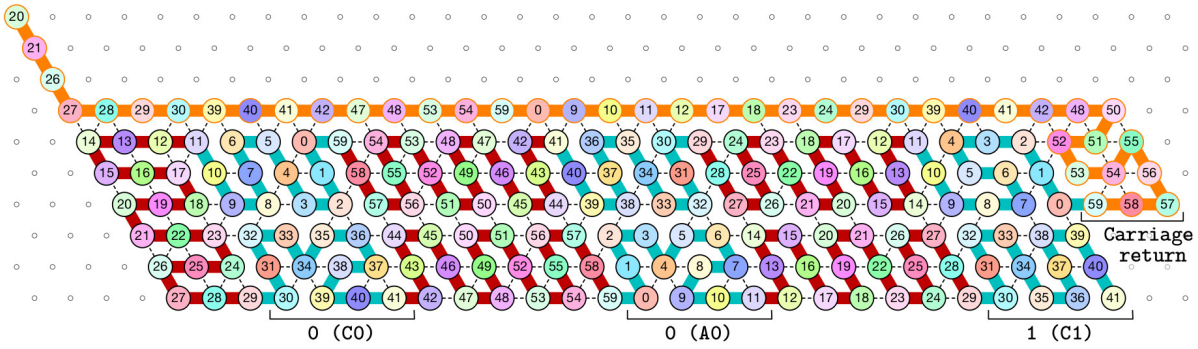


Figure 3.47: An Oritatami configuration simulating a binary counter, from [35].

3.6 Conclusion

Most of the classic work in tile self-assembly was done for the aTAM in the 2D lattice $\mathbb{Z}^2$. As we have seen, some works have explored self-assembly and related models in 3D, especially, in order to construct 3D objects such as polycubes, such as the FTAM, 3D aTAM, or DNA origami (Section 3.5). There has been some work on self-assembly on surfaces, mainly in the areas of shape replication from a theoretical perspective (Section 3.4.3), and self-assembled coatings in material sciences, from an experimental perspective (Section 3.4.4).

It is thus natural to further study tile self-assembly on 3D surfaces from a theoretical point of view. We will do this in the next chapters.

Chapter 4

Classification of flat surfaces using the aTAM

This chapter deals with the classification of a family of surfaces called *flat surfaces* using tile self-assembly. Flat surfaces are similar to the plane $\mathbb{Z}^2$, but they can be finite along one or two dimensions, and they wrap around themselves along their finite dimension(s) (in the first case they are called *torus* and in the latter case, *cylinder*). We will consider four types of flat surfaces, depending on which of their dimensions are finite. The way we want to classify these surfaces is to design a tile assembly system that can self-assemble on all these surfaces, and in which, for every kind of flat surface, some designated set of tile types appears in every terminal assembly if and only if the assembly takes place on a surface of that kind.

In Section 4.1, the types of flat surfaces that we consider are introduced: the flat torus, the flat vertical cylinder, the flat horizontal cylinder, and the plane $\mathbb{Z}^2$. In Section 4.2, the notion of aTAM on this family of surfaces is described. Finally, in Section 4.3, we present a TAS $\mathcal{S}_{\mathcal{T}}$ that is able to identify the flat surface on which the assembly is performed.

4.1 Flat surfaces

In this section, we define flat surfaces via the 2-dimensional integer lattice $\mathbb{Z}^2$. For a positive integer n, $\mathbb{Z}/n\mathbb{Z}$ is the additive group with elements $\{0, \ldots, n-1\}$. If $n = \infty$, we define $\mathbb{Z}/n\mathbb{Z}$ as $\mathbb{Z}$. Using these, we formalize the definition of the flat surfaces as follows.

Definition 4.1 (Flat surface)**.** *For $m, n \in \mathbb{N} \cup \{\infty\}$, the lattice $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ is called a* flat surface *and we denote it by $F_{m,n}$. Two positions (x_1, y_1) and (x_2, y_2) of $F_{m,n}$ are* adjacent *if (i) $|\min\{x_1, x_2\} - \max\{x_1, x_2\}| = 1 \bmod m$ and $y_1 = y_2$ or (ii) $x_1 = x_2$ and $|\min\{y_1, y_2\} - \max\{y_1, y_2\}| = 1 \bmod n$. For a position $p = (x, y)$ of $F_{n,m}$:*

- *the position $(x, (y+1) \bmod m)$ is the northern neighbor of p;*
- *the position $(x, (y-1) \bmod m)$ is the southern neighbor of p;*
- *the position $((x+1) \bmod n, y)$ is the eastern neighbor of p;*
- *the position $((x-1) \bmod n, y)$ is the western neighbor of p;*

For two integers $m, n \in \mathbb{N}$, the flat surface $F_{m,n}$ is a rectangular lattice of length m and height n on $\mathbb{Z}^2$. It is called a *flat torus.*

Definition 4.2 (Flat torus)**.** *The flat surface $F_{m,n}$ is called a* flat torus *when m, n are integers.*

The flat torus $F_{m,n}$ with $m, n \in \mathbb{N}$ when embedded in the plane $\mathbb{Z}^2$ is equivalent to a torus in $\mathbb{Z}^3$, by identifying parallel opposite sides without twisting, as shown in Figure 4.1.

Example 4.3. *The flat torus $F_{11,9}$ is presented in Figure 4.1. In $F_{11,9}$, the pink parallel sides are identified from west to east, and the blue parallel sides are identified from south to north.*

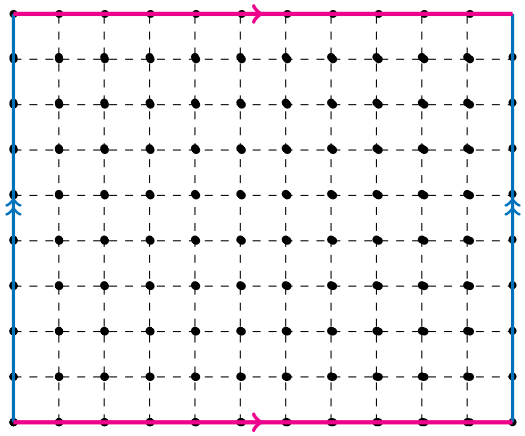


Figure 4.1: Representation of the flat torus $F_{11,9}$ in 2D.

Next, we consider the case where, for a flat surface $F_{m,n}$, exactly one of m and n is infinity. In this case, only two parallel sides are identified, and as before the identification is done in the same direction.

Definition 4.4 (Flat horizontal cylinder)**.** *The flat surface $F_{m,n}$ where m is ∞ and n in an integer, is called a* flat horizontal cylinder *and is denoted by $F_{\infty,n}$.*

Example 4.5. *The flat horizontal cylinder $F_{\infty,9}$ is depicted in Figure 4.2. The pink horizontal sides of $F_{\infty,9}$ are identified along the direction of west to east.*

The *flat vertical cylinder* is defined similarly to the flat horizontal cylinder.

Definition 4.6 (Flat vertical cylinder)**.** *The flat surface $F_{m,n}$, where m is an integer and n is ∞, is called a* flat vertical cylinder *and is denoted by $F_{m,\infty}$.*

Example 4.7. *The flat vertical cylinder $F_{11,\infty}$ is shown in Figure 4.3. The blue vertical parallel sides of $F_{\infty,9}$ are identified along the direction of south no north.*

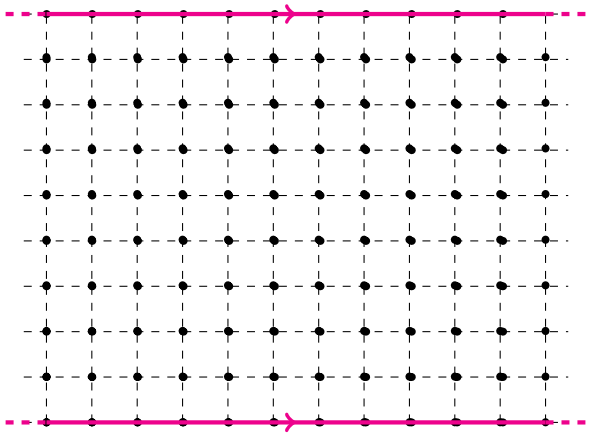


Figure 4.2: Representation of the flat horizontal cylinder $F_{\infty,9}$ in 2D.

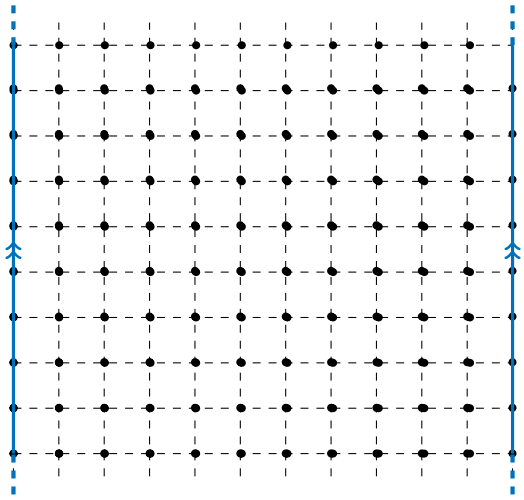


Figure 4.3: Representation of the flat vertical cylinder $F_{11,\infty}$ in 2D.

Remark 4.8. *The 2-dimensional discrete lattice $\mathbb{Z}^2$ corresponds to the flat surface $F_{m,n}$ when m and n both are infinity. See Figure 4.4 for an illustration.*

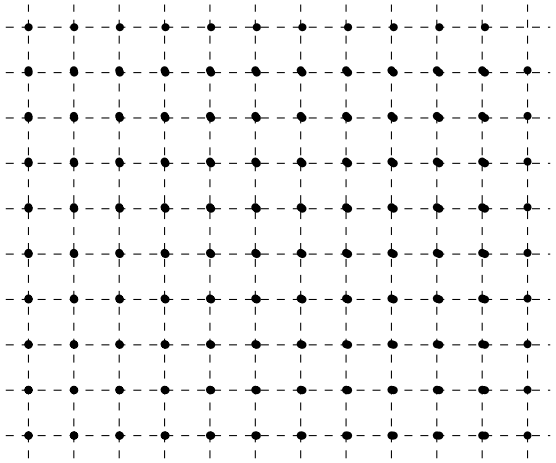


Figure 4.4: The 2-dimensional discrete lattice $\mathbb{Z}^2$ corresponds to the flat surface $F_{\infty,\infty}$.

Remark 4.9. *The flat surfaces $F_{m,n}$ and $F_{n,m}$ are isomorphic by the isomorphism i :*

$(x, y) \mapsto (y, x)$. *However, for a given position* (x, y) *of* $F_{m,n}$*, its northern neighbor (say) corresponds to the eastern neighbor of* $i(x, y) = (y, x)$ *in* $F_{n,m}$.

4.2 The aTAM on flat surfaces

The abstract Tile Assembly Model (aTAM) was presented in Section 3.3 on the plane $\mathbb{Z}^2$. In this section, we show that aTAM is applicable on $F_{m,n}$ for $m, n \in \mathbb{N} \cup \{\infty\}$. Indeed, the notions of north, east, south and west directions are consistent on those surfaces.

Proposition 4.10. *The aTAM is well-defined on flat surfaces.*

Proof. To be able to produce assemblies of an aTAM TAS $\mathcal{S}$, one needs the property that every position has four neighbors (north, east, south, west) and that the global north is well-defined. This is the case for all flat surfaces, with the directions as given in Definition 4.1. Thus, the statement follows. □

By Proposition 4.10, the notions for the aTAM on flat surfaces are derived in a natural way from the ones of the aTAM on the plane $\mathbb{Z}^2$. The tile types are the same.

Based on these explanations, next, we restate the formal definitions for the aTAM on flat surfaces. They are similar to the definitions of Chapter 3.3.2. Definition 4.11 is similar to Definition 3.25, Definition 4.12 is similar to Definition 3.28, Definition 4.13 is similar to Definition 3.30, and Definition 4.14 is similar to Definition 3.31. For infinite assemblies, Definition 3.33 is also valid for aTAM assemblies on flat surfaces.

A *position* (for a tile) is a couple (x, y) of $F_{n,m}$.

Definition 4.11 (aTAM tile assembly system on flat surfaces)**.** *An aTAM tile assembly system (TAS for short) on flat surfaces, is a quintuplet* $\mathcal{S} = (\Sigma, T, \alpha_\sigma, str, \tau)$*, where for every flat surface* $F_{m,n}$ *with* $m, n \in \mathbb{N} \cup \{\infty\}$*:*

- Σ *is an alphabet,*
- T *is a finite set of tile types,*
- τ *is a positive integer, the temperature,*
- *str is a function from* $\Sigma \cup \{\epsilon\}$ *to non-negative integers called* strength function *such that* $str(\epsilon) = 0$*, and*
- *The seed assembly* α_σ *is a set of tiles, each associated to a position of* $F_{n,m}$.

Definition 4.12 (Assembly on flat surface)**.** *Let* $F_{n,m}$ *be a flat surface, where* $m, n \in \mathbb{N} \cup \{\infty\}$.

An assembly α *of an aTAM TAS* $\mathcal{S} = (\Sigma, T, \sigma, str, \tau)$ *on* $F_{n,m}$ *is a partial function* $\alpha : F_{n,m} \dashrightarrow T$ *defined on at least one position* (x, y).

For two adjacent positions (x_1, y_1) *and* (x_2, y_2) *of* $F_{n,m}$ *with* $\alpha(x_1, y_1) = t$ *and* $\alpha(x_2, y_2) = t'$*, we say that* t *and* t' bind *if the glues of* t *and* t' *on the touching sides of* t *and* t' *are equal and have positive strength.*

We recall that the *assembly graph* G_α associated to α (defined in Definition 3.29) has as its vertices, the placements of Pl(C) that have an image by α, and two placements p and p' are adjacent in G_α if the tiles $\alpha(p)$ and $\alpha(p')$ bind.

Definition 4.13 (τ-stable assembly on flat surface)**.** *For an integer* τ *(usually, the temperature of the TAS), an assembly* α *is* τ-stable *if for breaking* G_α *into several connected components by removing any set* C *of edges of* G_α*, the sum of strengths of the bonds corresponding to edges of* C *is at least* τ.

Definition 4.14 (Producible and terminal assemblies on flat surfaces)**.** *Let $\mathcal{S} = (\Sigma, T, \alpha_\sigma, str, \tau)$ be a TAS and F be a flat surface. An assembly α of S is producible on F if either α is the seed assembly, or if α can be obtained from a producible assembly β by τ-stably adding a single tile on F.*

We write $\beta \to^{\mathcal{S}} \alpha$ when α is producible from β and we show the set of producible assemblies of $\mathcal{S}$ on F by $A^F[\mathcal{S}]$.

A producible assembly α is the result *of an* assembly sequence *$\alpha_0 \to \alpha_1 \to \ldots \to \alpha$, where $\alpha_0 = \alpha_\sigma$.*

A (possibly infinite) producible assembly is a terminal assembly if no tile can be τ-stably attached to it. The set of producible, terminal assemblies of $\mathcal{S}$ is denoted by $A^F_\square[\mathcal{S}]$.

Remark 4.15. *Note that every flat surface is invariant under translation. Thus, the set of producible assemblies of any aTAM TAS is independent of the position of the seed (up to translation of the whole assembly).*

4.3 Classifying flat surfaces using the aTAM

We now have the necessary definitions for presenting the main result of this chapter: classifying flat surfaces via an aTAM tile assembly system. In this TAS, two special tile types, x and y, distinguish these four surface types. Figure 4.5 displays a Venn diagram for such a tile assembly system and its two distinguishing tile types. In this illustration, the blue circle represents the flat surfaces that contain the tile type x in all terminal assemblies, and the pink circle represents the ones containing the tile type y.

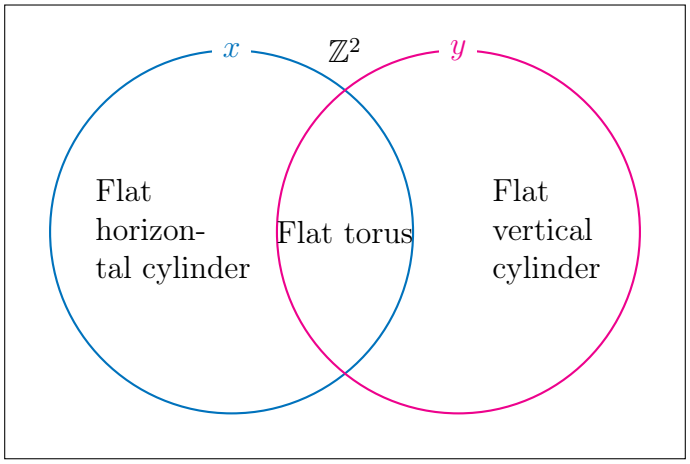


Figure 4.5: The Venn diagram that illustrates the presence of the two distinguishing tile types x and y that identify the flat surfaces. The blue circle represents the flat surfaces that contain the tile type x in all the terminal assemblies, and the pink circle represents the ones containing the tile type y. The flat vertical cylinder contains x and not y, the flat horizontal cylinder contains y and not x, the flat torus contains both of them, and the plane $\mathbb{Z}^2$ contains none of them.

Theorem 4.16. *There exists a TAS $\mathcal{S}_\mathcal{F} = (\Sigma, T, \alpha_\sigma, str, \tau)$ with tile types $\{x, y\} \subset T$ such that on every flat surface $F = F_{m,n}$ with $m, n \in \mathbb{N} \cup \{\infty\}$ and $m, n \geq 3$, the following holds:*

1. *if $m \in \mathbb{N}$, for every terminal assembly $\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$, $y \in \alpha$. Otherwise (if $m = \infty$), there is no $\alpha \in A^F[\mathcal{S}_\mathcal{F}]$ such that $y \in \alpha$, and*

2. *if $n \in \mathbb{N}$, for every terminal assembly $\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$, $x \in \alpha$. Otherwise (if $n = \infty$), there is no producible assembly $\alpha \in A^F[\mathcal{S}_\mathcal{F}]$ such that $x \in \alpha$.*

Moreover, the terminal assemblies of $\mathcal{S}_\mathcal{F}$ are unique, and unless both $m, n \in \mathbb{N}$, the assemblies of $\mathcal{S}_\mathcal{F}$ leading to a (limiting) terminal assembly are infinite.

The tile assembly system $\mathcal{S}_\mathcal{F} = (\Sigma, T, \sigma, str, \tau)$ is as follows:

- $\Sigma = \{h'', hn, hs, v'', ve, vw\}$ *is the alphabet,*
- T *is the finite set of tile types* $\{\sigma, v, h, x, y\}$. *See Figure 4.6 for a representation of the tile types.*
- α_σ *takes a single tile of type* $\sigma = (\epsilon, h'', \epsilon, v'')$, *placed at* $(0, 0)$.
- $str : \Sigma \cup \{\epsilon\} \longrightarrow \{0, 1, 2\}$ *such that* $str(\epsilon) = 0$, $str(h'') = str(v'') = 2$ *(the double quotes " refer to their strength 2) and all other values are 1.*
- $\tau = 2$

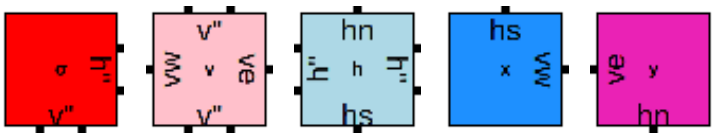


Figure 4.6: The tile types of the aTAM TAS $\mathcal{S}_\mathcal{F}$ for classification of flat surfaces. The seed is in red.

Proof. The process of growing a producible assembly $\alpha \in A^F[\mathcal{S}_\mathcal{F}]$ on a flat surface $F = F_{m,n}$ with $m, n \in \mathbb{N} \cup \{\infty\}$ and $m, n \geq 3$, starts once the seed $\sigma = (\epsilon, h'', \epsilon, v'')$ is placed at $(0, 0)$. Afterwards, a horizontal ribbon of tile types of $h = (hn, h'', hs, h'')$, and a vertical ribbon of tile types of $v = (v'', ve, v'', vw)$ from south and east of the seed grow respectively via the strength 2-labels of h'' and v'', as shown in Figure 4.7. This is the case because no other tile type can be attached to the seed.

Let m be a finite positive integer and $\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$ be a terminal assembly of $\mathcal{S}_\mathcal{F}$ on $F_{m,n}$. In every terminal assembly, the horizontal ribbon comes back to the west of the seed, by wrapping around the surface of $F_{m,n}$ using $m - 1$ tiles of type h. As a result, the tile type $x = (hs, vw, \epsilon, \epsilon)$ appears in α at this new collision of horizontal ribbon and vertical ribbon, as shown in Figure 4.8. Due to the assumption that $m \geq 3$, there is enough space for this collision and for the tile type x there. The tile of type x is added to the assembly α via label hs between the south of h and north of x, and via label vw between the west of v and the east of x.

Now, let $m = \infty$ and $\alpha \in A^F[\mathcal{S}_\mathcal{F}]$ be a producible assembly of $\mathcal{S}_\mathcal{F}$ on $F_{m,\infty}$. Considering that the eastern label of x is wv, a tile of type x can appear in the assembly only on the western side of the ribbon of tiles of type v (the only tile type of the system with label vw at its west side). Thus, this happens only when some tiles of type h appear west of the seed. However, when $m = \infty$, the growth of the tiles of type h continues indefinitely towards the east and the tiles of type h never appear at the west of the seed. As a result, when $m = \infty$, no tile of type x can appear in an assembly of $\mathcal{S}_\mathcal{F}$ on $F_{m,n}$.

The proof is similar for n and the tile of type y.

Let n be a finite number and $\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$ be a terminal assembly of $\mathcal{S}_\mathcal{F}$ on $F_{m,n}$. In every terminal assembly, the vertical ribbon comes back to the north of the seed by wrapping around the surface of $F_{m,n}$. Consequently, a tile of type $y = (\epsilon, \epsilon, hn, ve)$ appears

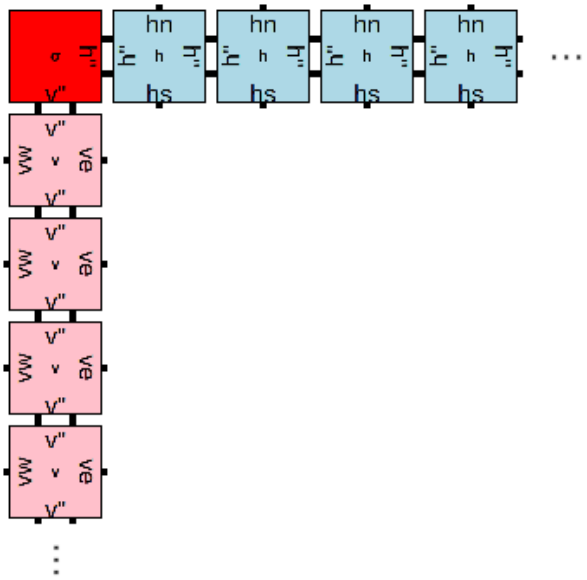


Figure 4.7: An assembly $\alpha \in A^F[\mathcal{S}_\mathcal{F}]$ on the plane $\mathbb{Z}^2$. α starts by the seed σ, and vertical and horizontal ribbons of tile types v and h grow respectively from the south and the east of σ.

in the assembly at the new collision between the vertical ribbon and horizontal ribbon, as shown in Figure 4.8. Due to assumption that $n \geq 3$, there is enough space for this collision and for the tile of type y to appear there. A tile of type y is added to the assembly α via label hn between the north of h and south of y, and label ve between the east of v and west of y.

Now, let $n = \infty$ and $\alpha \in A^F[\mathcal{S}_\mathcal{F}]$ be a producible assembly of $\mathcal{S}_\mathcal{F}$ on $F_{m,\infty}$. Since the south label of y is hn, a tile of type y appears in the assembly only on the north side of the ribbon of tiles of type h. This happens only when some tiles of type h appear in the positive positions of the assembly. However, when $n = \infty$, the growth of the tiles of type v continues indefinitely towards the south and the tiles of type v never appear north of the seed. As a result, when $n = \infty$, no tile of type y can appear in an assembly of $\mathcal{S}_\mathcal{F}$ on $F_{m,n}$.

Finally, it follows from the tile types that no terminal assembly other than the ones described above are possible on flat surfaces. In the case of a torus, the terminal assembly is finite. In the other cases, the assembly sequence leading to a terminal assembly is infinite, and we have a limiting terminal assembly. □

For the TAS $\mathcal{S}_\mathcal{F}$ introduced in Theorem 4.3, the presence of tile types x and y in the terminal assemblies on a given flat surface identifies its type. In fact, the tile type x appears in the assembly if and only if m is finite. In other words, the flat surface is either the flat horizontal cylinder $F_{m,\infty}$, or the torus $F_{m,n}$. In addition, the tile type y appears in the assembly if and only if n is finite i.e. if the flat surface is either the flat vertical cylinder $F_{\infty,n}$, or the torus $F_{m,n}$. The main result is stated as follows:

Corollary 4.17 (Classification of flat surfaces by an aTAM TAS)**.** *Let $\mathcal{S}_\mathcal{F} = (\Sigma, T, \sigma, str, \tau)$ (with two tile types $\{x, y\} \subset T$) be the tile assembly system presented in Theorem 4.3. For every flat surface $F = F_{m,n}$ with $m, n \geq 3$, the following holds:*

– *F is a flat vertical cylinder if and only if x appears in every terminal assembly*

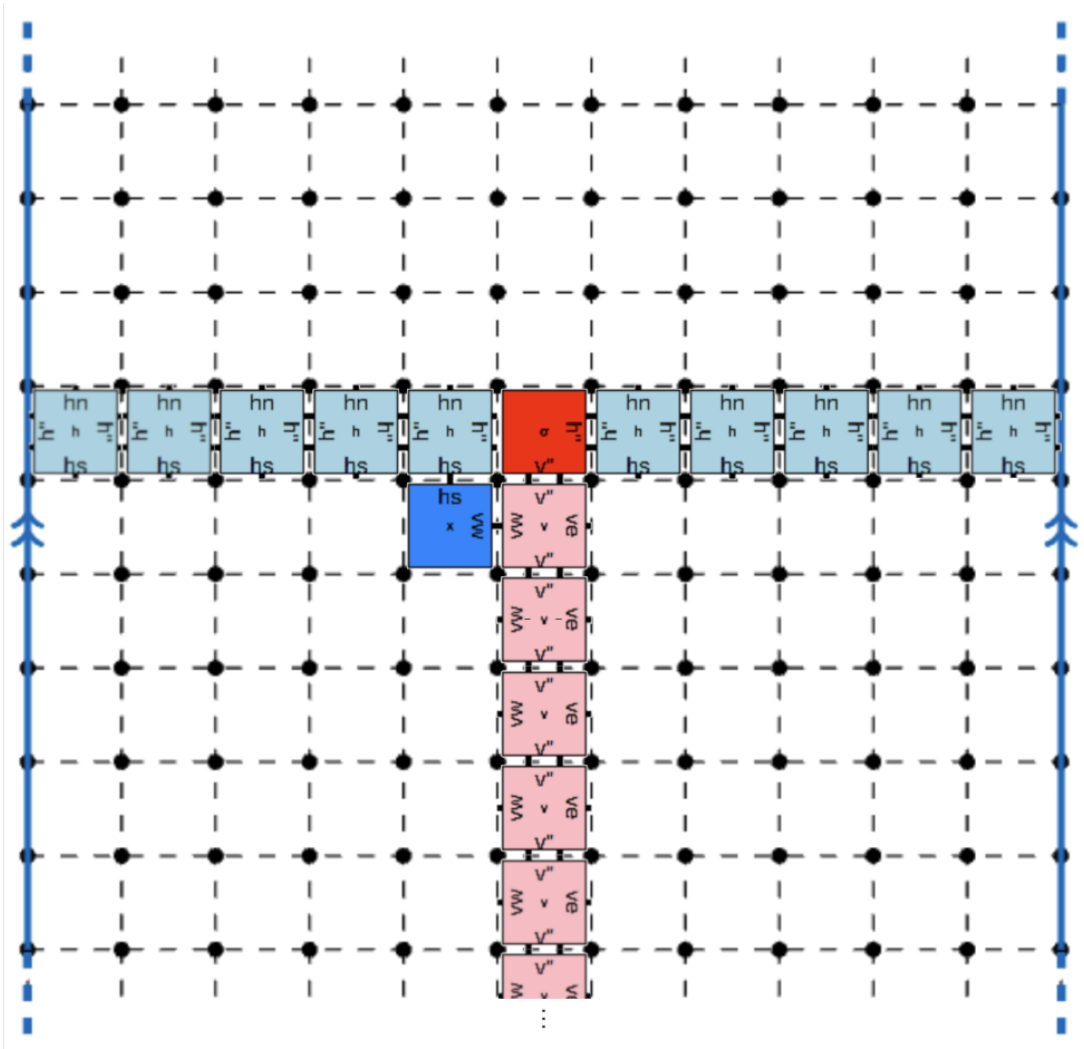


Figure 4.8: An assembly $\alpha \in A^F[\mathcal{S}_F]$ on a flat surface when $m = 11$ and $n = \infty$. α starts by the seed σ, and a vertical ribbon of tiles of type v grow from the south of σ. Also, a horizontal ribbon of tiles of type h grows from the east of σ. After adding 5 tiles of type h, the tiles reach the eastern side of this figure and continue from the western side. When the ribbon of tiles of type h arrives to the west of the seed, x appears there, between tiles of types h and v. The assembly on the figure is the terminal limiting assembly (assuming the vertical ribbon is infinite).

$\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$ *on* F *and* y *does not;*

- F *is a flat horizontal cylinder if and only if* y *appears in every terminal assembly* $\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$ *on* F *and* x *does not,*
- F *is a flat torus if and only if both* x *and* y *appear in every terminal assembly* $\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$ *on* F*, and*
- F *is the plane* $\mathbb{Z}^2$ *if and only if none of* x *and* y *appear in any terminal assembly* $\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$ *on* F*.*

Proof. Let $F = F_{m,n}$ be a flat surface with $m, n \geq 3$ and $\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$ on F. Firstly, F is a flat vertical cylinder if and only if $m \in \mathbb{N}$ and $n = \infty$. Thus, according to Theorem 4.16, F is a flat vertical cylinder if and only if the tile type x appears in α and x does not. See Figure 4.8 for an illustration.

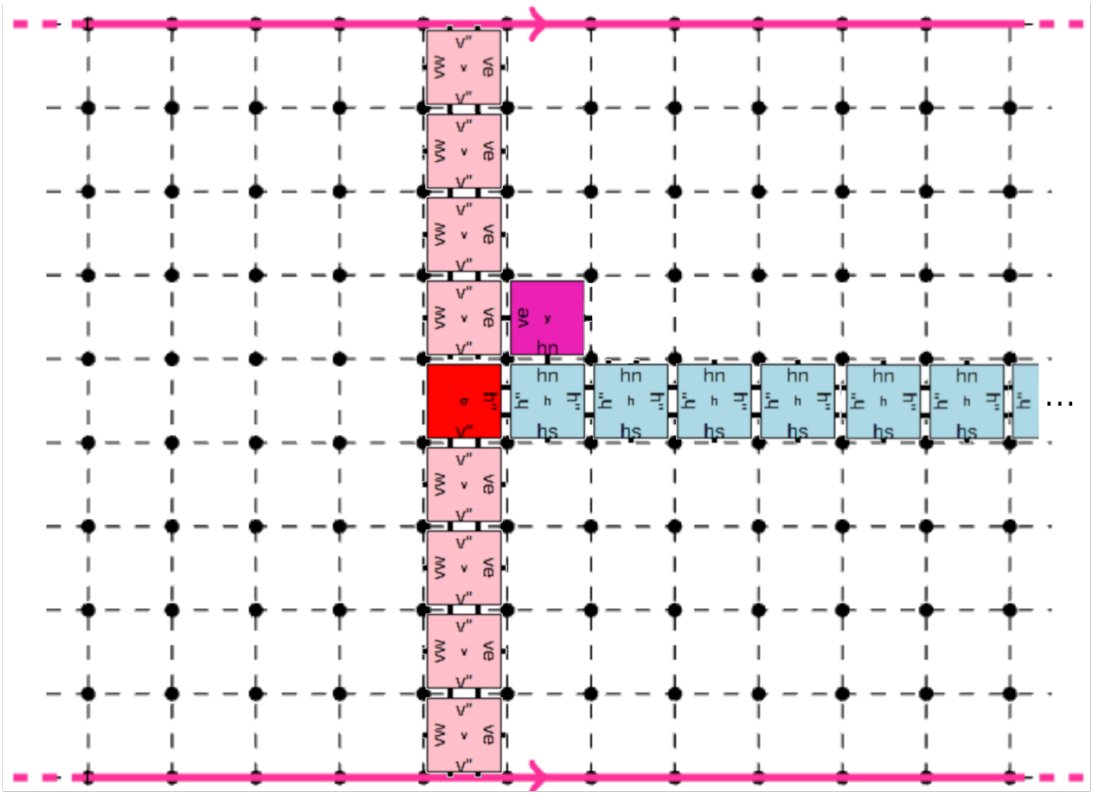


Figure 4.9: An assembly $\alpha \in A^F[\mathcal{S}_F]$ on a flat surface with $m = \infty$ and $n = 9$. α starts by the seed σ, and a horizontal ribbon of tiles of type h grows from the south of σ. A vertical ribbon of tiles of type v grows from the east of σ. After adding four tiles of type v, the tiles arrive to the southern side of the figure and α continues from the northern side. When the ribbon of tiles of type v collides with the north of the seed, y appears, adjacent to tiles of both types h and v. The assembly on the figure is the terminal limiting assembly (assuming the horizontal ribbon is infinite).

Secondly, F is a flat horizontal cylinder if and only if $m = \infty$ and $n \in \mathbb{N}$. Thus, according to Theorem 4.16, F is a flat horizontal cylinder if and only if the tile type y appears in α and x does not. See Figure 4.9 for an illustration.

Next, F is a flat torus if and only if $m \in \mathbb{N}$ and $n \in \mathbb{N}$. Thus, according to Theorem 4.16, F is a flat torus if and only if both tile types x and y appear in α. See Figure 4.10 for an illustration.

Lastly, F is the plane $\mathbb{Z}^2$ if and only if $m = \infty$ and $n = \infty$. Thus, according to Theorem 4.16, F is $\mathbb{Z}^2$ if and only if none of the tile types x and y appear in α. See Figure 4.7 for an illustration. □

4.4 Concluding remarks

Corollary 4.17 obtained in this chapter implies that via the TAS $\mathcal{S}_\mathcal{F}$, we are able to classify the type of flat surface on which the assembly takes place: if a terminal assembly $\alpha \in A^F_\square[\mathcal{S}_\mathcal{F}]$ contains a tile of type x or y, then we are not on $\mathbb{Z}^2$. If it contains both x and y, then we are on a finite torus. If α contains x but not y, the flat surface is a flat vertical cylinder and if it contains y but not x, it is a flat horizontal cylinder. For further

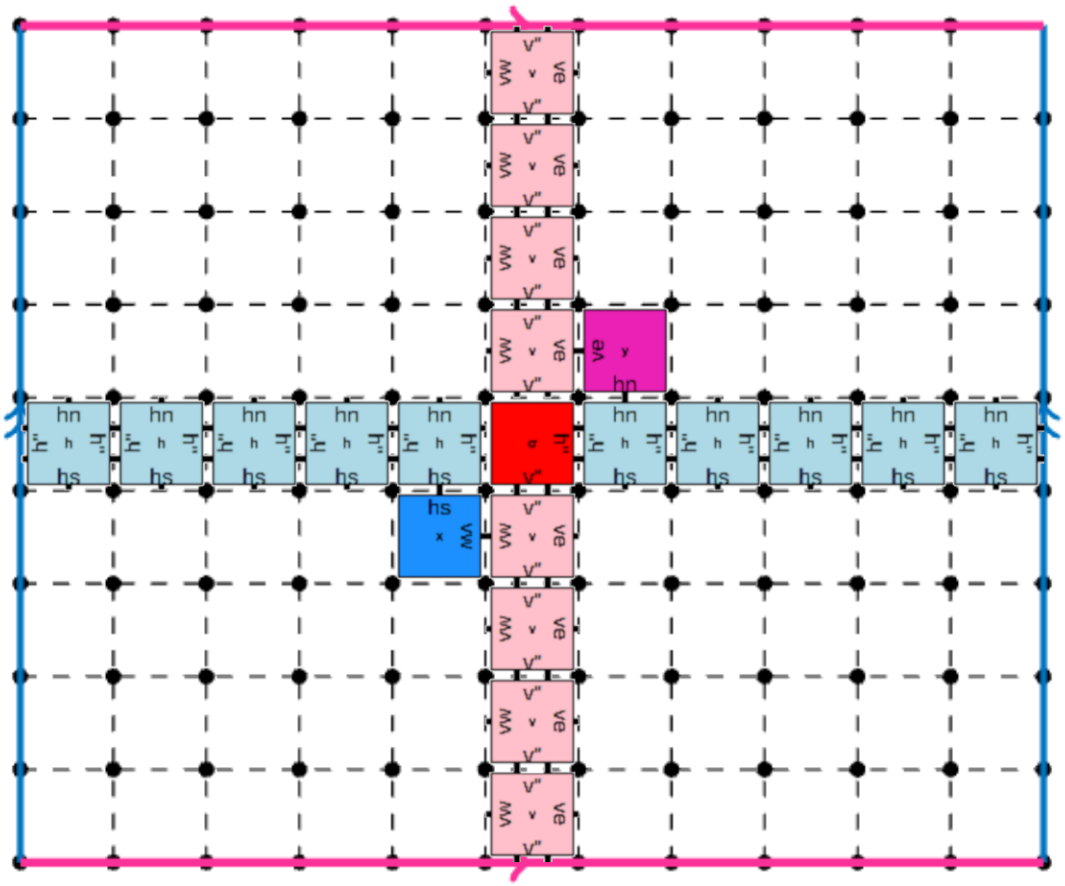


Figure 4.10: An assembly $\alpha \in A^F[\mathcal{S}_F]$ on a flat torus $F_{9,11}$. α starts by the seed σ, and vertical and horizontal ribbons of tile types v and h grow respectively from the south and the east of σ. After adding 6 tiles of type h, the horizontal ribbon reaches the eastern side of $F_{9,11}$ and α continues from its western side. After placing again 5 tiles of type h, the ribbon of tiles of type h collides with the west of the seed. At this new junction between tiles of types h and v, a tile of type x appears. Moreover, After adding 5 tiles of type v, the tiles reach the southern side of $F_{9,11}$ and α continues from the northern side of $F_{9,11}$. After placing 4 tiles of type v, the ribbon of tiles of type v collides with the north of the seed. At this new junction between tiles of types h and v, a tile of type y appears. The assembly on the figure is the terminal assembly.

illustration, Table 4.11 summarizes this classification.

As noted in Remark 4.9, the two flat surfaces $F_{m,n}$ and $F_{n,m}$ are isomorphic, however, if $m \neq n$, for a given aTAM TAS, their assemblies are not necessarily the same, since their global north is along a dimension of different length. This is because tiles are not allowed to rotate in the aTAM, unlike in the FTAM described in Chapter 3.5.

In flat surfaces, the assemblies are invariant under translations. However, this may not be the case if the surface is irregular, for example for a polycube. Our main goal is to extend the type of classification obtained in this chapter to more complicated types of surfaces. We will tackle the case of simple polycubes of genus at most 1 in Chapter 7. Before this, we have to introduce an extension of the aTAM that works on more complex surfaces, in Chapter 5.

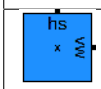
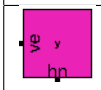
	Flat torus	Flat vertical cylinder	Flat horizontal cylinder	$\mathbb{Z}^2$
	✓	✗	✓	✗
	✓	✓	✗	✗

Table 4.11: The table of presence of tile types x (in blue) and y (in pink) in terminal assemblies of $\mathcal{S}_{\mathcal{F}}$ on flat surfaces.

Chapter 5

The SFTAM: Surface Flexible Tile Assembly Model

In this chapter we present the new *Surface Flexible Tile Assembly Model* or SFTAM. Our goal of introducing the SFTAM is to enable tile self-assembly using classic unit square tiles on 3D surfaces more complex than the 2D plane. In Chapter 4, we have designed a TAS that distinguishes between the four types of flat surfaces. Our goal is to obtain a similar result for more complicated types of surfaces, in particular, determine the genus of certain polycube surfaces. However, as such surfaces can be irregular, the aTAM is not suitable. Thus, we need to introduce an extension of the aTAM suitable for our study. We believe that our model could be useful in practical scenarios where we have a host surface on which we want to perform assemblies. As seen in Chapter 3.4, previous work (like the FTAM) has mostly focused on *creating* certain 3D shapes using self-assembly, in particular polycubes, or, assembly was performed on rather restricted surfaces like mazes.

The SFTAM can be seen as an intermediate between aTAM and FTAM. Unlike in the FTAM, our aim for introducing the SFTAM is *not* for building 3D structures or surfaces: we assume that the host surface already exists. The SFTAM assemblies resemble those of the aTAM, and we avoid the difficulty of creating shapes one is faced with in the FTAM.

Polycubes are a natural option as the first choice to do tile self-assembly in 3-dimensional space. Indeed, as the faces of polycubes are made of unit squares, we can easily place on them classic square tiles as the ones used in the aTAM. The main difference with the aTAM will be that the bonds between tiles bend on the edges of polycubes.

We first give the basic necessary notions and the formal definition of the SFTAM in Section 5.1. We give some examples in Section 5.2. In Section 5.3, we compare the SFTAM with some previous models. Finally, we conclude in Section 5.4.

5.1 The definition of SFTAM

In this section, we formally define the Surface Flexible Tile Assembly Model, SFTAM. In classical models like the aTAM, the assemblies are on the plane $\mathbb{Z}^2$. Here, we work on 3-dimensional surfaces. To upgrade tile self-assembly on 3-dimensional surfaces using rigid tiles, it is mandatory that the bonds between them be flexible so that the assemblies grow on edges of surfaces and beyond.

The definitions of this section are adapted from the definitions for the aTAM in Chapter 3.3.2. Definition 5.1 is similar to Definition 3.24, Definition 5.4 is similar to Definition 3.25, Definition 5.6 is similar to Definition 3.28, Definition 5.7 is similar to Definition 3.30, and Definition 5.8 is similar to Definition 3.31.

Like in the aTAM, tiles are $2D$ unit squares whose sides are assigned the labels of the tile type, but there are some differences in the definition.

Definition 5.1 (Tile type in the SFTAM)**.** *Let Σ be a finite label alphabet and ϵ represent the null label. A* tile type *t is a 4-tuple $t = (t_1, t_2, t_3, t_4)$ with $t_i \in \Sigma \cup \overline{\Sigma} \cup \{\epsilon\}$ for each $i = \{1, 2, 3, 4\}$, where ϵ is the* null label*. Each copy of a tile type is a* tile*. By analogy with the aTAM, t_1, t_2, t_3 and t_4 are called the* northern label*,* eastern label*,* southern label *and* western label*, respectively. Each label $l \in \Sigma \cup \overline{\Sigma}$ will receive a* strength *$str(l) \in \mathbb{N}$, with $str(l) = str(\bar{l})$.*

Remark 5.2. *In the SFTAM, tiles are allowed to translate and rotate (unlike in the aTAM), but do not reflect as in the FTAM. To prevent two tiles of the same type attaching each other on the same glue after a 180 degree rotation of one of them, we have added the complementary glue labels in order to control the assembly better (this is also present in the FTAM [30]).*

Figure 5.1 shows the four possible orientations of a single tile of type $t = (a, b, c, d)$ with $str(a) = 2$ and $str(b) = str(c) = str(d) = 1$.

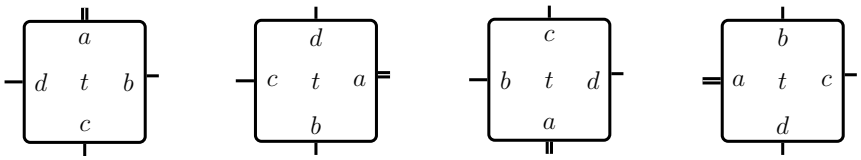


Figure 5.1: The four possible orientations of a tile of type t in the SFTAM.

We now formally define the way we can place a tile on the surface of a polycube. This definition is a simlification of the one used in the FTAM (Definition 3.42).

Definition 5.3 (Placement in SFTAM)**.** *Let C be a polycube. A* placement *$p = (f, o)$ on C consists of a facet f on the surface of C, and a side o of f, called its* orientation*. We denote the set of all placements in C by $Pl(C)$.*

An example of a facet on a polycube and the four possible placements on it are shown in Figure 5.2.

Note that for a given facet of a polycube, there are four possibilities for placing a given tile for each orientation. Given a tile type $t = (t_1, t_2, t_3, t_4)$ and a placement $p = (f, o)$, *placing* t at the placement p defines a mapping from the edges of f to the glues of t : the i-th side of f (starting from the orientation o and going clockwise, looking from the exterior of the surface of C) is associated with t_i.

Now we can define a *tile assembly system* in the SFTAM.

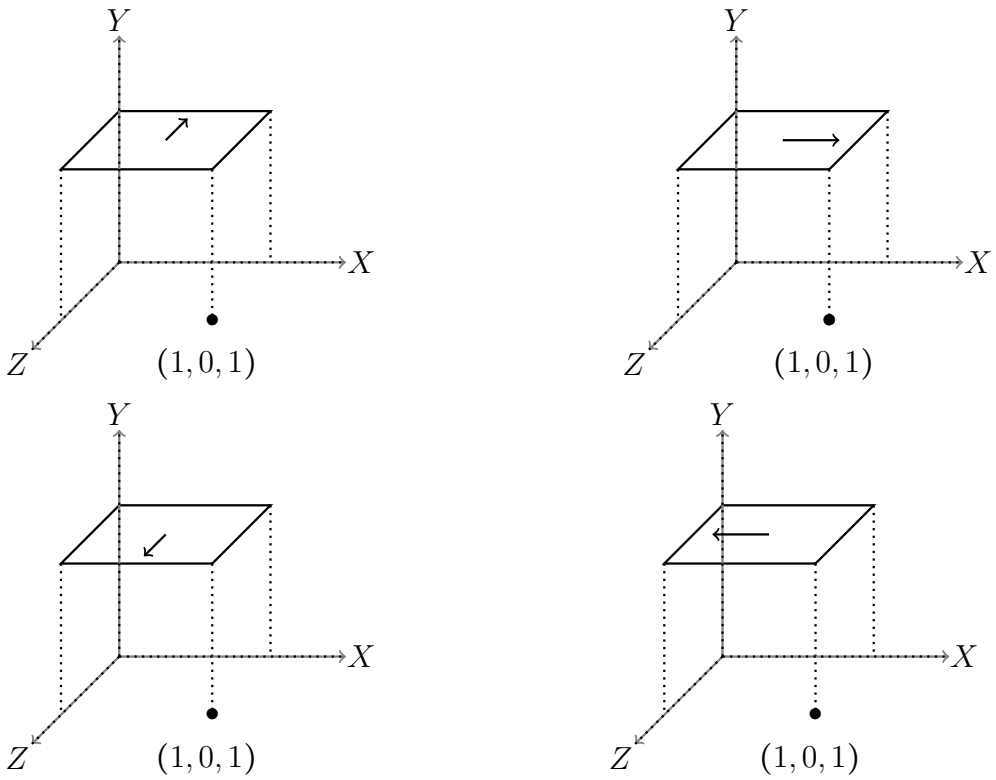


Figure 5.2: A facet f of a polycube and the four possible placements of a tile on it. For a tile of type $t = (t_1, t_2, t_3, t_4)$, the arrow indicates the side o of f where the side with glue t_1 of t will be placed; this corresponds to the placement (f, o).

Definition 5.4 (Tile assembly system (TAS) in the SFTAM over a surface)**.** *A* tile assembly system*, or TAS, over the surface of a given polycube C is a quintuple $\mathcal{S} = (\Sigma, T, \alpha_\sigma, str, \tau)$, where:*

- *Σ is an alphabet;*
- *T is a finite set of tile types on Σ;*
- *the* seed assembly *α_σ is a set of tiles, each with a specified placement of $Pl(C)$;*
- *str is a function from $\Sigma \cup \{\epsilon\}$ to non-negative integers called* strength function *such that $str(\epsilon) = 0$ and $str(t_i) = str(\bar{t}_i)$ for $t_i \in \Sigma$, and;*
- *$\tau \in \mathbb{N}$ is called the* temperature*.*

Remark 5.5. *As in Remark 3.27, although the formal definition asks that the seed assembly gives a fixed set of placements, in practice we might have a TAS where the seed assembly can vary, i.e., it belongs to a family of possible placements. Moreover, even for the same seed configuration, the exact placement on the polycube might also vary.*

Next, we present some further notations. The following definitions of the assembly dynamics are almost the same as the ones of the aTAM given in Chapter 3.3.2, with some necessary modifications (as noted in Remark 5.2, binding is done for *complementary* labels).

Definition 5.6 (Assembly in the SFTAM)**.** *An* assembly *α of a SFTAM TAS $\mathcal{S} = (\Sigma, T, \alpha_\sigma, str, \tau)$ on a polycube C is a partial function $\alpha : \mathrm{Pl(C)} \dashrightarrow T$ defined on at least one placement and such that for each facet f of C, there is at most one placement (f, o) where α is defined.*

For placements $p = (f, o)$ and $p' = (f', o')$ of Pl(C) *with $\alpha(p) = t$ and $\alpha(p') = t'$ such that f and f' are distinct but have a common side s, we say that t and t'* bind *if the glues l of t and l' of t' placed on s are* complementary *(i.e. $l = \overline{l'}$) and have positive strength.*

We recall that the *assembly graph* G_α associated to α (defined in Definition 3.29) has as its vertices the placements of Pl(C) that have an image by α, and two placements p and p' are adjacent in G_α if the tiles $\alpha(p)$ and $\alpha(p')$ bind. Recall the following definition.

Definition 5.7 (τ-stable assembly)**.** *For an integer τ (usually, the temperature of the TAS), an assembly α is* τ-stable *if for breaking G_α into several connected components by removing any set C of edges of G_α, the sum of strengths of the bonds corresponding to edges of C is at least τ.*

We start with a seed and then we add tiles one by one. This is formally described as follows.

Definition 5.8 (Producible and terminal assemblies in the SFTAM)**.** *Let C be a polycube and $\mathcal{S} = (T, \alpha_\sigma, \tau)$ a SFTAM TAS with α_σ positioned on C. An assembly α of $\mathcal{S}$ is* producible *on C if either α is α_σ where α_σ is placed on a set of facets of C, or if α can be obtained from a producible assembly β by τ-stably adding a single tile from T on C.*

We write $\beta \to^{\mathcal{S}} \alpha$ when α is producible from β and we denote the set of producible assemblies of $\mathcal{S}$ by $A^C[\mathcal{S}]$.

An assembly is terminal *if no tile can be τ-stably attached on C. The set of producible, terminal assemblies of $\mathcal{S}$ is denoted by $A^C_\square[\mathcal{S}]$.*

5.2 Examples and remarks

We now give an example of a TAS in the SFTAM.

Example 5.9 (SFTAM TAS)**.** *Let $\mathcal{S} = (\Sigma, T, \alpha_\sigma, str, \tau)$ be the SFTAM TAS where:*

- *$\Sigma = \{a, b, c\}$ is the alphabet;*
- *$T = \{t_0, t_1, t_2\}$ with $t_0 = (d, a, b, c)$, $t_1 = (\overline{b}, \epsilon, \epsilon, \overline{a})$ and $t_2 = (\epsilon, \overline{c}, \overline{d}, \epsilon)$ are the tile types;*
- *the seed assembly α_σ is a single tile of type t_0 placed on a placement;*
- *$str(\epsilon) = 0$, $str(a) = str(\overline{a}) = 1$ and $str(b) = str(\overline{b}) = str(c) = str(\overline{c}) = 2$;*
- *the temperature $\tau = 2$.*

The tile types of the TAS $\mathcal{S}$ from Example 5.9 are depicted in Figure 5.3, and an assembly is shown in Figure 5.4.

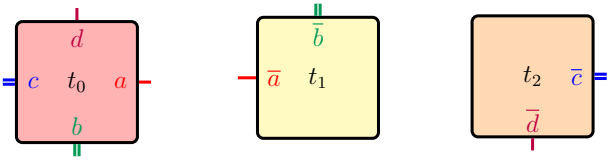
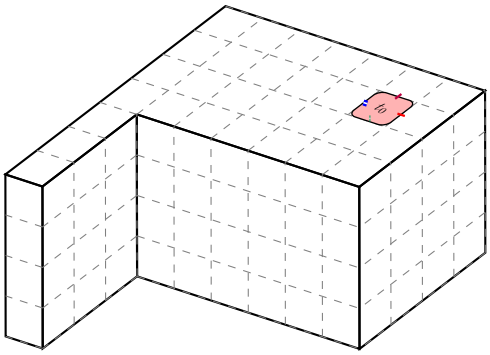
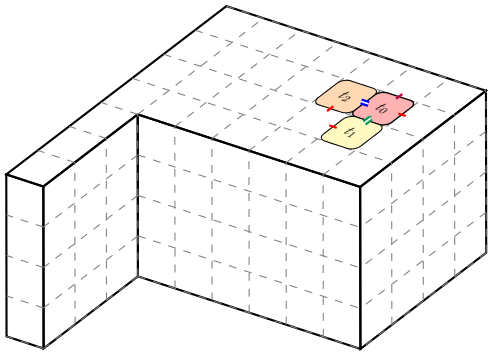


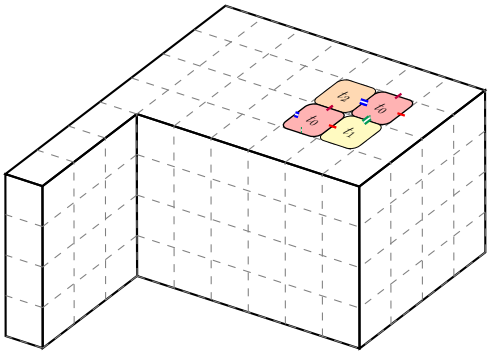
Figure 5.3: The tile types of the SFTAM TAS $\mathcal{S}$ from Example 5.9.



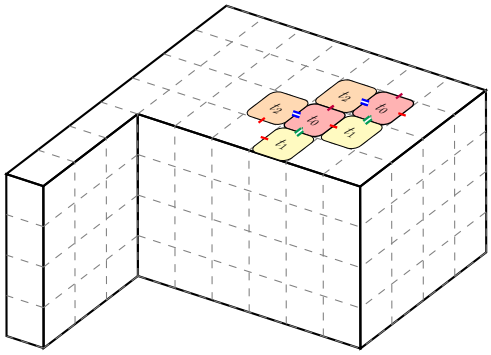
(a) First stage of the assembly: the seed is placed.



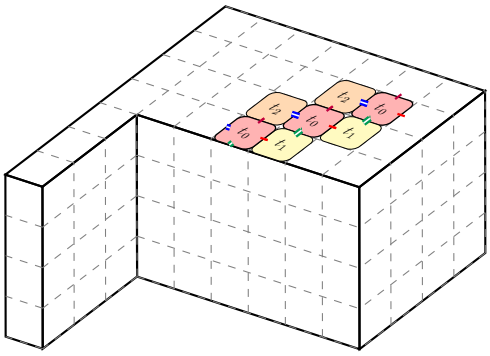
(b) Second stage of the assembly, after addition of two tiles of type t_1 and t_2.



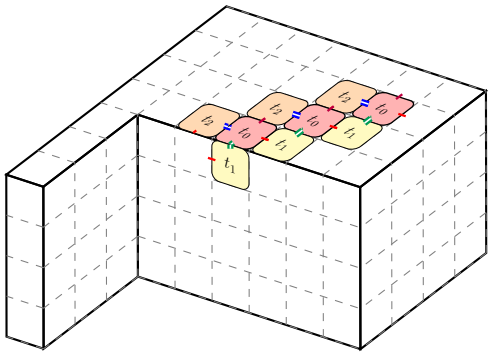
(c) Third stage of the assembly: one tile of type t_0 is added.



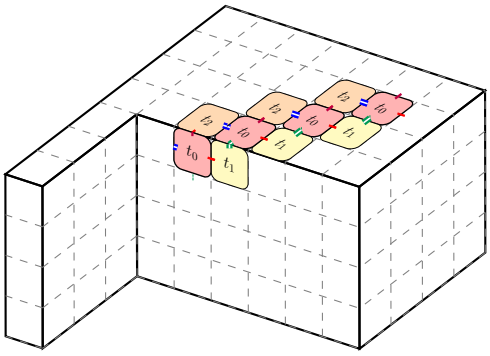
(d) Fourth stage of the assembly: two tiles of type t_1 and t_2 are added.



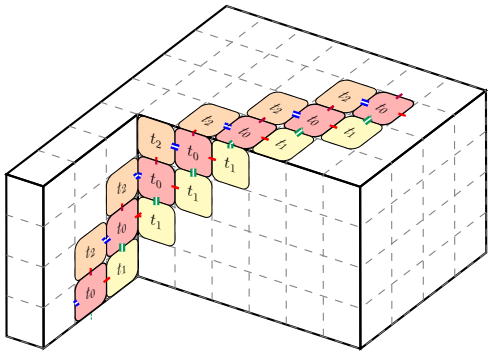
(e) Fifth stage of the assembly: one tile of type t_0 is added.



(f) Sixth stage of the assembly: two additional tiles of types t_1 and t_2 are placed.



(g) Seventh stage of the assembly.

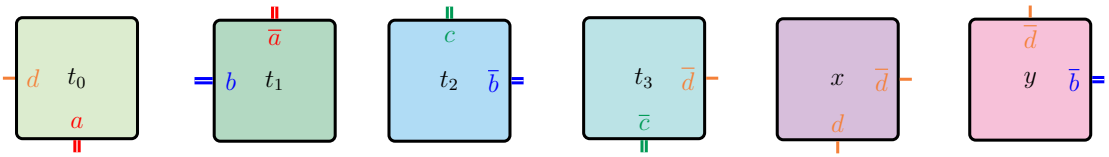


(h) Advanced stage of the assembly.

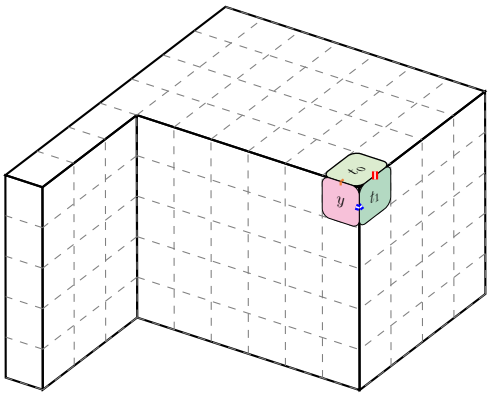
Figure 5.4: An assembly of the SFTAM TAS $\mathcal{S}$ from Example 5.9.

Remark 5.10. *For a given polycube C and an SFTAM TAS $\mathcal{S}$, for different placements of the seed, the sets of producible assemblies of $\mathcal{S}$ on C can be different. The SFTAM TAS $\mathcal{S}_1$ with tiles presented in Figure 5.5(a) and the tile type t_0 as seed is an example of such a SFTAM system in temperature 2. See Figure 5.5(b)–(e) for an illustration of the situations that by changing the placement of the seed, the assembly can change.*

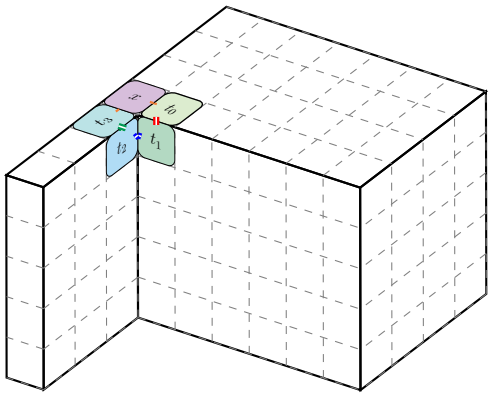
Moreover, for two different polycubes, the sets of producible assemblies for the same TAS can change too: consider two polycubes each having a type of vertex not present on the other polycube (for example, a vertex incident with five facets, like in Figure 5.5(b) may or may not be present). They give rise to different topologies of the facet graphs, and to assemblies that are not producible on the other surface.



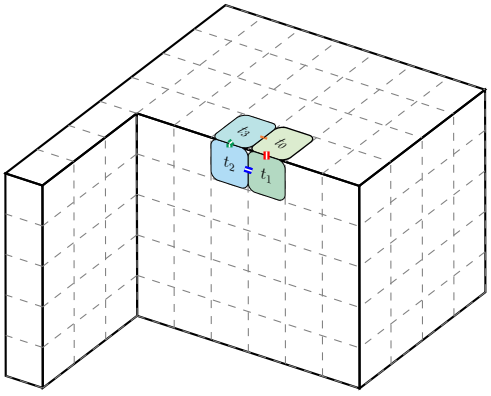
(a) Tile types of the SFTAM TAS $\mathcal{S}_1$.



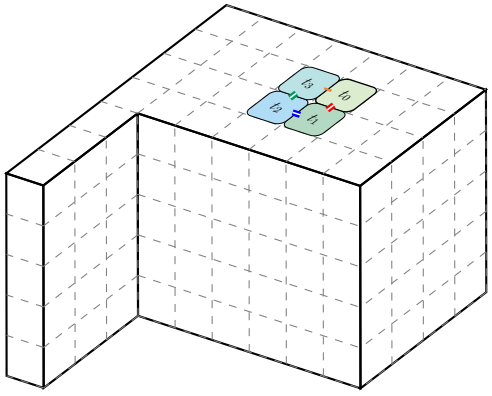
(b) On a convex corner. The order of placement of the tiles is first t_0, then t_1, and finally y.



(c) On a concave corner. The order of placement of the tiles is first t_0, then t_1, t_2, t_3, and then x.



(d) On an edge. The order of placement of the tiles is first t_0, then t_1, t_2 and finally t_3.



(e) On a face. The order of placement of the tiles is first t_0, then t_1, t_2 and finally t_3.

Figure 5.5: Illustration of Remark 5.10: four different assemblies of a TAS $\mathcal{S}_1$ seeded at different placements of a polycube.

5.3 Comparison and connections with previous models

5.3.1 Assemblies of aTAM systems on polycubes

In the SFTAM, unlike in the aTAM, the tiles are allowed to rotate. This is because on a 3D surface, the "global north" is not well-defined. For example, recall that is Chapter 4, the aTAM assemblies on the two flat surfaces $F_{n,m}$ and $F_{m,n}$ for $m \neq n$ can be different. This would not be the case in the SFTAM. In some special cases however, the assemblies can corrrespond in the aTAM and SFTAM.

Note that an assembly of an aTAM TAS $\mathcal{S}$ can take place on a polycube surface C in the SFTAM (by considering $\mathcal{S}$ as an SFTAM TAS), as long as there is enough space on C. This is formalized in the following lemma.

Lemma 5.11. *Let α be a producible assembly of an aTAM TAS $\mathcal{S}$ on $\mathbb{Z}^2$, and let C be a polycube. If there exists a function $i : Dom(\alpha) \to C$ such that i taken as a function of the assembly graph G_α to the subgraph of the facet graph $G_f(C)$ induced by $Im(i)$ is a graph isomorphism, then the image of α under i is producible on C in the SFTAM (after a suitable relabeling of the glues of $\mathcal{S}$).*

Proof. One may need to relabel the glues to obtain an SFTAM TAS from $\mathcal{S}$: if there are glues that are used both on north/south and east/west tile sides in $\mathcal{S}$, they can be renamed to give two distinct glues. Moreover, if there is a glue label, say l, on the west side of a tile, it should be renamed as $\overline{l}$ on the east sides of tiles, so that binding is possible in the SFTAM.

If a tile of the seed assembly of $\mathcal{S}$ is placed at a point p_s in $\mathbb{Z}^2$, it is placed at $i(p_s)$ on C. Since the tile bonds can fold along edges of C, the assembly on C proceeds exactly as it proceeds on $\mathbb{Z}^2$, and each tile placed at a point p in $\mathbb{Z}^2$ is placed at point $i(p)$ on C. □

Instead of using Lemma 5.13, we may use a stronger assumption using the following definition.

Definition 5.12 (Underlying rectangle)**.** *Given an aTAM assembly α on $\mathbb{Z}^2$, we call the smallest axis-parallel rectangle containing the assembly, its* underlying rectangle*.*

If an SFTAM assembly is on a 3D surface, it is permitted to fold along the tiles' edges. It is then possible to generalize the underlying rectangle to SFTAM assemblies as the smallest subset R of the surface which contains the assembly and such that the subset of the facet graph $G_f(C)$ induced by R is isomorphic to a rectangular grid graph, if it exists. With this definition, we can reformulate a weaker version of Lemma 5.11 as follows.

Lemma 5.13. *Let α be a producible assembly of an aTAM TAS $\mathcal{S}$ on $\mathbb{Z}^2$ with underlying rectangle R, and let C be a polycube. If there exists a function $i : R \to C$ such that i taken as a function of the subgraph of the 2D lattice induced by R to the subgraph of the facet graph $G_f(C)$ induced by $Im(i)$ is a graph isomorphism, then the image of α under i is producible on C in the SFTAM (after a suitable relabeling of the glues of $\mathcal{S}$).*

5.3.2 Producible SFTAM assemblies on polycubes may not be producible in $\mathbb{Z}^2$

For a given TAS S in a SFTAM and a polycube C, the assemblies of S in C may not correspond to the ones in $\mathbb{Z}^2$ for the corresponding aTAM system. There are several

reasons for this. Firstly, tiles can rotate in the SFTAM, which is not possible in the aTAM. Nevertheless, one could introduce three additional tile types for each tile type of S (to simulate the ability to rotate tiles) and argue that the producible assemblies of S in the SFTAM can also be producible in the aTAM. However, a more serious obstacle exists: the fact that the polycubes have corners. Indeed, as seen in Section 3.2.3, the vertices of the polycubes may be incident with up to six facets of the surface in complicated ways, while in $\mathbb{Z}^2$, every vertex is adjacent to exactly four facets. See Figure 5.5(b) for a simple example of an assembly that needs three mutual adjacent placements, something that is not possible in $\mathbb{Z}^2$.

5.3.3 Comparison of SFTAM with FTAM

In the SFTAM, unlike in the FTAM, the tiles do not reflect. Indeed, the tiles stick to a given polycube and the normal vector n of the placement in the FTAM is not needed. The reason is that we fix that the normal vector (as used in the FTAM) so that it always starts inside of the polycube and points to the outside of the structure. So, the cyclic order of the tiles' sides is uniquely determined by the orientation of its placement. Moreover, we can assume that the tiles have an inner face and an outer face, and that they always attach with the inner face in contact with the surface. Hence, we do not need to consider a normal vector for the placement of a tile. However, if we would consider non-orientable surfaces, then we would still need a normal vector since inside and outside of the surface is not well-defined.

Another essential point is that if we are able to obtain a surface via the FTAM, it is possible to do SFTAM tile self-assembly on it. Thus, one could combine the FTAM and SFTAM to first produce a 3D surface via the FTAM and then run a SFTAM assembly on the surface.

5.4 Concluding remarks

We remark that the SFTAM can be used on surfaces other than polycubes. In fact, on any orientable quadrangulated surface, where the quadrangles are all unit squares, one can use the SFTAM. For example, this is the case of the flat surfaces studied in Chapter 4.

In experimental studies, as tiles are built from DNA strands (which are not rigid), it is difficult to precisely control the shapes of the tiles. Hence, one can also imagine a scenario where the tiles are stretchable, and in this case, any quadrangulated surface would be admissible for the SFTAM.

The absence of the normal vector in the definition of the SFTAM works well with orientable polycubes/surfaces. If we were working on a non-orientable surface, we would need a normal vector for the placement of tiles.

Chapter 6

Finding the middle of a track in the aTAM or SFTAM

The aim of this chapter is to define a SFTAM system that can be used to find the middle of a track on a polycube. Informally speaking, a *track* on a polycube is a set of facets on the surface of the polycube that, when unfolded and embedded in $\mathbb{Z}^2$, forms a rectangle. When two of its parallel sides are distinguished as *Start* and *Finish*, it is called an *explicitly bounded track*. Consider Figure 6.1(b) for an illustration, where p (in green) is the track on the polycube with the start and finish points, and the explicitly bounded track is seen in dark gray. The width of the explicitly bounded track needs to be at least $3\log(N)$, where N is its height, in order to have enough space for our construction.

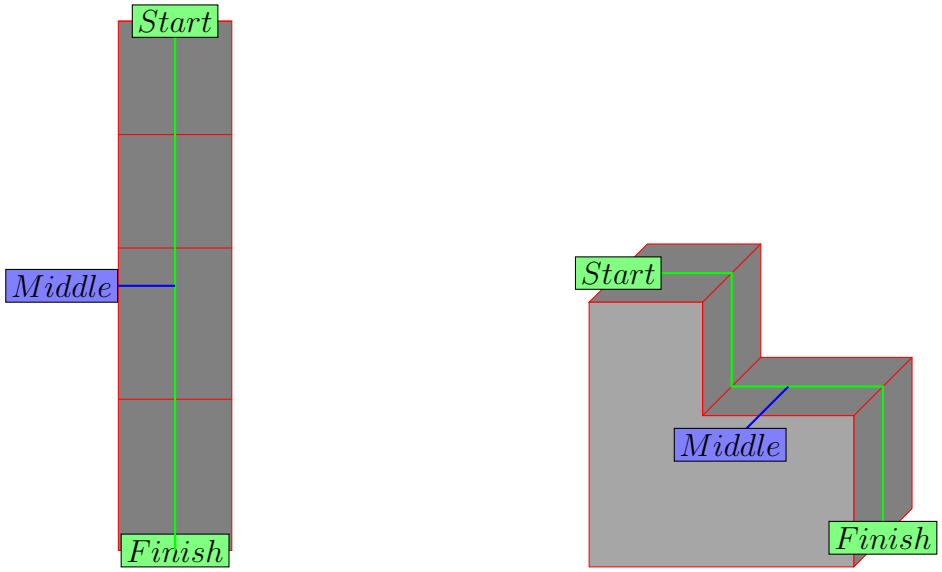
In order to find the middle of p, we present an aTAM TAS $S_{1/2}$ named *middle finding system* such that if an assembly of $S_{1/2}$ grows from a seed in the start point, a particular tile t_m sits only in the middle of the track and not anywhere else.

According to Lemma 5.13, for the underlying rectangle R of an aTAM assembly α in $\mathbb{Z}^2$, if we can find a track of the same structure as R on the surface of a polycube C, then we can produce the same assembly on C in the SFTAM. (Up to renaming of labels to prevent tiles to rotate and attach differently as they should, as described in the proof of Lemma 5.11.) Hence, in this chapter we only work with tracks and underlying rectangles on $\mathbb{Z}^2$ in the aTAM, but the results can be applied to tracks and underlying rectangles on polycubes.

Finding the middle of a track on a polycube will be an essential tool for the constructions developed in Chapter 7. Moreover, this construction is general enough that it can be useful in different contexts.

We first present some aTAM TAS's that will be used as building blocks for the middle finding system. They are based on binary counters used to measure the length of the track. By Lemma 5.13, we can assume that we are on $\mathbb{Z}^2$, and our results will also hold for tracks on polycubes.

The outline of this chapter is as follows. First in Section 6.1 we start with some preliminaries. Next, in Section 6.2 we introduce the *Increasing Binary Counter* (Lemma 6.5) and in Section 6.3, the *Decreasing Binary Counter* (Lemma 6.6). In Section 6.4, we present a TAS called *U-turn system* (Lemma 6.7). Then, we present the main result of this section, the *middle finding system lemma* (Lemma 6.8) in Section 6.5 to find the middle of a track. We conclude in Section 6.6.



(a) in $\mathbb{Z}^2$ (b) on a polycube

Figure 6.1: Finding the middle of a track on a polycube. The track is made of the facets that are in the dark grey area.

6.1 Preliminaries

We start by defining some essential notions for this chapter.

One essential point is how to represent a number via tiles. This is classic in the literature of tile self-assembly, see for example the binary counter from [65] presented in Example 3.35. To this aim, we use the binary representation of numbers and we associate a number to a row of tiles such that every bit of the number is assigned to a tile with the appropriate label.

Definition 6.1 (Row tile number)**.** *Let T_0 and T_1 be two sets of tiles with labels 0 and 1, respectively. Let N be an integer and $a_1...a_2a_n$ be its binary representation with $n = \lceil log_2 N \rceil$. A row of tiles with labels $a_1^*, a_2, ..., a_n$ is the* row tile number *representation of N such that the distinct tile a_1^*, represents the most significant bit of the number.*

Example 6.2. *See an example of the row tile number of the number* 12 *in Figure 6.2.*

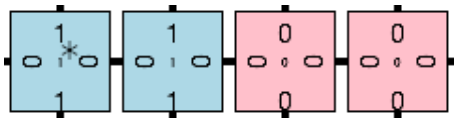


Figure 6.2: Representation of the number 12 by its row tile number.

We now formalize the notion of a track.

Definition 6.3 (Track)**.** *Given a polycube C, a* track *is a set F of facets of the surface of C such that the subgraph of the facet graph $G_f(C)$ induced by the facets in F is isomorphic to a rectangular grid graph.*

See Figure 6.1 for an example of a track on a polycube.

In order to do finite tile self assembly or block the growth of an assembly inside a restricted area, we define *explicitly bounded tracks* as follows.

Definition 6.4 (Explicitly bounded track)**.** *An* explicitly bounded track *R (on $\mathbb{Z}^2$ or on a polycube) is a track with a partial assembly consisting of two rows of tiles placed on two opposite sides of R.*

See Figure 6.1 for a representation of an explicitly bounded track.

The tiles that border an explicitly bounded track can be considered part of the seed of an assembly on this track.

6.2 The increasing binary counter system

We will use the tile representation of numbers in binary introduced in Section 6.1 to read or measure the length of a track on the underlying rectangle of an assembly. To do so, we introduce specific *counter systems*.

We now introduce the *Increasing Binary Counter system* (IBC), which is similar to the binary counter TAS from Example 3.35 from [65] (with the main difference that our counter is bounded, while that from [65] is unbounded). First, we explain what the IBC system is, then we present its tile types. Afterwards, we show how it works by explaining the assembly process.

In the production of the IBC system, the tiles of each row (excluding the tiles σ_+ and t_{++}) form a row tile number representing the index of the row in which they are located. Consequently, whenever the assembly stops by getting to the finish point of a track, the length of the assembly corresponds to the last row tile number. This number is a measure of the length of the underlying rectangle of the assembly.

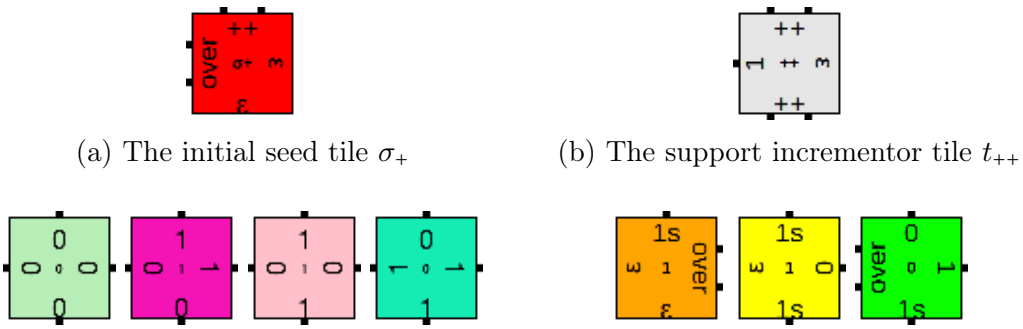
Lemma 6.5 (Increasing binary counter system, IBC)**.** *The* increasing binary counter system *or* IBC system*, is a SFTAM tile assembly system over $\mathbb{Z}^2$ such that if the bottom of the seed is placed at the origin, each row (excluding the tiles of type σ_+ and t_{++}) denotes its position in binary on the y axis and the leftmost tiles of all rows whose number is 2^k for some $k \geq 1$ is of type t_O.*

The system is described by a quintuple $\mathcal{S} = (\Sigma, T, \alpha_\sigma^I, str, \tau)$, where:

- $\Sigma = \{0, 1, 1_s, ++, over, \epsilon\}$
- $T = \{\sigma_+, t_{++}, t_{0+0}, t_{0+1}, t_{1+0}, t_{1+1}, t_S, t_O, t_{OS}\}$

- α_σ^I *places a tile of type* $\sigma_+ = (++, \epsilon, \epsilon, over)$ *(the* ϵ *labels can be replaced by other labels depending on the situation) together with a column of* support tiles *of arbitrary size, placed on the north side of* σ_+*, all of whose west labels are* 1 *(the other labels can be arbitrary).*
- $str : \Sigma \cup \{\epsilon\} \longrightarrow \{0, 1, 2\}$ *such that* $str(\epsilon) = 0$*. The values of the str function for* Σ *are* $str(0) = str(1) = str(++) = str(1_s) = 1$ *and* $str(++) = str(over) = 2$*.*
- $\tau = 2$

The IBC tile types are shown in Figure 6.3.



(a) The initial seed tile σ_+

(b) The support incrementor tile t_{++}

(c) The rule tile types t_{0+0}, t_{0+1}, t_{1+0} and t_{1+1}

(d) The tile types t_O, t_S and t_{OS} related to the most significant bit

Figure 6.3: The IBC tile types.

Proof. We start by describing the IBC tiles.

The seed of the IBC.

There is one tile type $\sigma_+ = (++, \epsilon, \epsilon, over)$, and one incrementor tile type $t_{++} = (++, \epsilon, ++, 1)$ (the gray tile in Figure 6.3) called the support tiles. Together, one tile of type σ_+ with a column of tiles of type t_{++} form the block seed of the IBC. Note that the tile type of the support is allowed to be different from t_{++}, as long as its western labels are 1.

The rule tile types of IBC.

There are also seven sum rule tile types: four "standard" rule tile types and three rule tile types related to the most significant bit. However, from now on when we mention rule tiles, we refer to standard rule tiles.

First, we explain how the binary sum is done by these tiles. In each rule tile type, the inputs are the labels at the east and south of the tile, for example a and b. The two outputs are: the classic binary sum $a + b$ in the northern label, and the carry bit, which is displayed on the western label of the tile. The rule tile types of the IBC are: $t_{0+0} = (0, 0, 0, 0), t_{0+1} = (1, 0, 1, 1), t_{1+0} = (1, 1, 0, 0)$ and $t_{1+1} = (0, 1, 1, 0)$ with the side labels as mentioned, and where the inner label is the value of the sum, the same as the northern label.

The most significant bit tile types of the IBC.

There are three tile types for the most significant bit. They use the same logic as the rule tile types. The tile type $t_S = (1_S, 0, 1, \epsilon)$ represents the most significant bit of the row number when that number is not a power of 2. In case $a = b = 1$ in the currently most significant bit, there is an overflow after the sum operation $a + b$. In this case, the two rule tiles t_{OS}, t_O are used to change the most significant bit. The tile $t_{OS} = (0, 1, 1_s, over)$

is the connecting tile between t_O and t_S in case of the overflow and $t_O = (1_s, over, \epsilon, \epsilon)$ is the new most significant bit when the row number is a power of 2.

The process of the assembly in the IBC TAS.

The process starts with the seed. The tiles of the support start the process of incrementing the row below.

A tile of type t_O binds to the west of the seed tile by label "over" with strength 2 in the east. Note that the label of t_O is 1, which corresponds to row number 1, where it is located. Now the incrementation process starts. Each tile is bound by its eastern and southern neighbors, i.e. for having a tile t_* with the label $a+b$, there is a tile at the south with north label a and there is a tile at the east with west label b. The first tile to bind in each row is the easternmost, it binds with the support. The carry bit of the sum appears in the west side of the tile t_*. The northern label of t_* is $a+b$ too. See an example of an assembly for the number 16 in Figure 6.4.

Note again that each row represents the row number in which it is located along the y axis. This process continues for the whole height of the support of α_σ^I.

In the terminal assembly, the top row is the row tile number N, where N is the length of the support. Moreover, the tiles of type t_O only appear at the left of a row when the row is a power of 2, and the highest one is the row number $\max\{c \mid 2^c \leq N\}$. □

The features of our IBC System regarding previous models

Other binary counters were introduced in the literature, for example the one presented in Example 3.35 from [65]. However, the IBC system has some specific features that are necessary in the construction of the middle finding system. Among them, we do not need the support tiles at the bottom of the production. More remarkable, there is no infinite (or arbitrarily large) line of 0 tiles at the left of the binary number of each row. Thus, the counter does not fill the whole plane and instead, we have a counter ribbon of tiles as a production. In consequence, a production of the IBC System is able to *measure* the length of a track on a given polycube due to Lemma 5.13.

6.3 The decreasing binary counter system

The *decreasing binary counter system* is a tile assembly system that implements a reverse counter.

This counter starts with row tile number N, and decreases until the $(N+1)$st row representing row tile number $2^{\lceil \log(N) \rceil} - 1$, where a special tile (of type t^*_{1-0}) is placed at the left. Exactly one tile of this type is present in the terminal assembly, indeed, for all the next rows, the assembly uses other tile types for the most significant bit and cycles until it reaches an obstacle. This special tile can then be used by a later assembly, for example as a seed.

Lemma 6.6 (Decreasing Binary Counter system, DBC system)**.** *The* Decreasing binary counter system *or* DBC system *for short, is a tile assembly system over* $\mathbb{Z}^2$ *defined by a quintuple* $\mathcal{S} = (\Sigma, T, \alpha_\sigma^D, str, \tau)$*, where:*

- $\Sigma = \{0, 0', 1, 1', --, \epsilon\}$
- $T = \{\sigma_-, t_{--}, t_{0-0}, t_{0-1}, t_{1-0}, t_{1-1}, t^*_{0-0}, t^*_{0-1}, t^*_{1-0}, t^*_{1-1}\}$*, see Figure 6.5.*
- *the seed assembly* α_σ^D *is made of a binary row tile number* N*, a tile of type* $\sigma_- = (--, \epsilon, \epsilon, 0)$ *(where the* ϵ *labels are arbitrary) at the east of it, and at the north of*

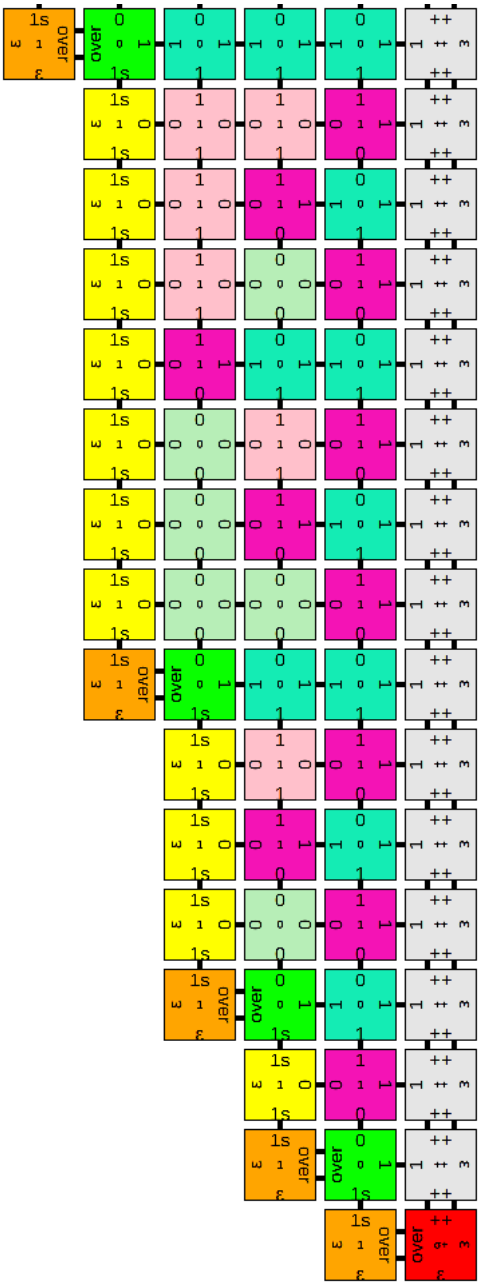


Figure 6.4: Assembly of row tile number 16 in the IBC.

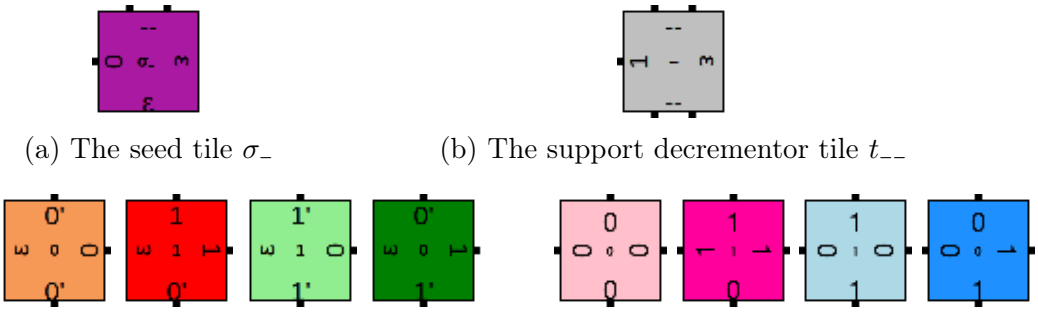
this tile, a column of support tiles that have western label 1*, of arbitrary size (for example of type* t_{--}*, but they can be of other types) placed on the north of the* σ_{-} *tile. The column has height at least* N*.*

- $str : \Sigma \cup \{\epsilon\} \longrightarrow \{0,1,2\}$ *such that* $str(\epsilon) = 0$*,* $str(--) = 2$ *and all other values are 1.*
- $\tau = 2$

The terminal assembly of the DBC system is a rectangle with exactly one tile of type t^*_{1-0} *located at the left of the* $(N+1)$*st row (which corresponds to row tile number* $2^{\lceil \log(N) \rceil} - 1$*).*

Proof. The DBC tiles are shown in Figure 6.5.

The seed of the DBC.



(a) The seed tile σ_{-} (b) The support decrementor tile t_{--}

(c) The tiles t^*_{0-0}, t^*_{0-1}, t^*_{1-0}, and t^*_{1-1} (d) The tiles t_{0-0}, t_{0-1}, t_{1-0} and t_{1-1}

Figure 6.5: The DBC tile types.

An assembly of the DBC TAS starts when a binary row number binds to the decrement operation tile of type $\sigma_D = (--, \epsilon, \epsilon, 0)$ to form a DBC seed for the decrementation operation.

There is a decrementor tile of type $t_{--} = (--, \epsilon, --, 1)$ which triggers the decrementation of the row below it into a new row to the left of the next tile of type t_{--}.

The rule tile types of the DBC.

There are eight rule tiles, four for the most significant bit and four for the others. The inputs are the labels of the east and south of the tile, for example a and b, and there are two outputs. The first one, that is located in the northern label, is the result of the binary subtraction $a - b$. The second output is the western label of the tile such that its value is needed to borrow for the next operation. The label of the new tile itself is also $a - b$. With this description, the rule tiles are $t_{a-b} = (a - b, b, a, borrow)$, i.e. $t_{0-0} = (0, 0, 0, 0)$, $t_{0-1} = (0, 1, 1, 1)$, $t_{1-0} = (1, 0, 1, 0)$, $t_{1-1} = (1, 1, 0, 0)$, $t^*_{0-0} = (0', 0, 0', \epsilon)$, $t^*_{0-1} = (0', 1, 1', \epsilon)$, $t^*_{1-0} = (1', 0, 1', \epsilon)$, $t^*_{1-1} = (1', 1, 0', \epsilon)$. See Figure 6.6 for an illustration.

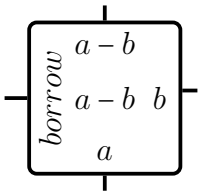
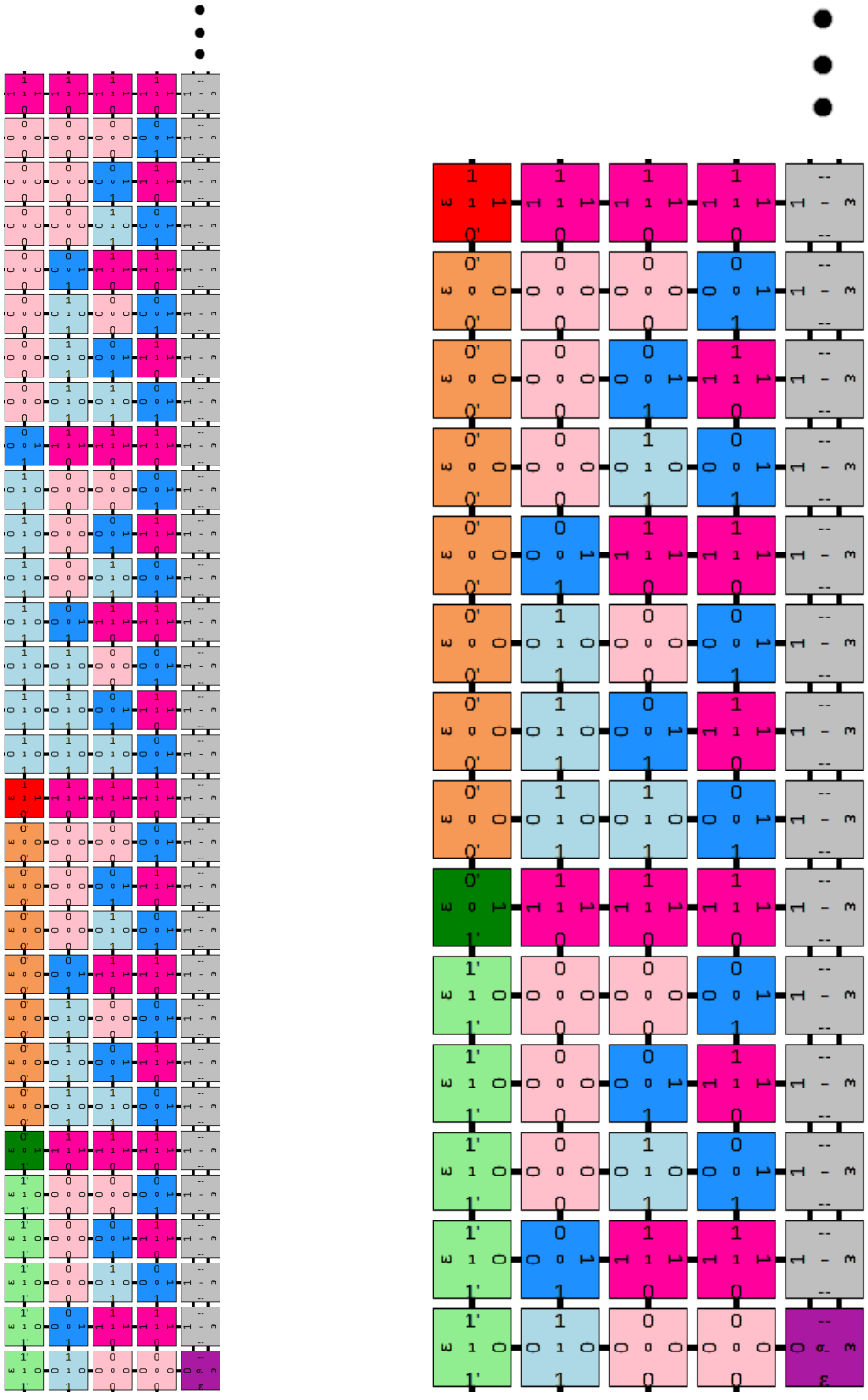


Figure 6.6: DBC rule tiles

The process of the assembly in the DBC TAS.

The counter starts at an arbitrary value N and grows along a vertical column of support tiles. A decrementation process starts by relying on the support tiles that trigger the decrementation of the row below it into a new row to the left of the support tile. Each row i shows the number $N - i$ modulo $min\{2^k \mid N \leq 2^k\}$. Indeed, when the row's value counts down to 0, the value of the next row will be the bits' maximum possible capacity of the number that the process was started with. Moreover, note that the most significant bit tiles are the tiles of types t^*_{0-0}, t^*_{0-1}, t^*_{1-1} and the row of the number 0 for the first time is exposed whenever a tile t^*_{1-0} appears in the most significant bit in the row after the one of 0, i.e. the row of -1. An example of a DBC assembly for $N = 12$ is presented in Figure 6.7, where the tile of type t^*_{1-0} is presented in red. After the appearance of t^*_{1-0}, the tiles of types t^*_{0-0}, t^*_{0-1}, t^*_{1-0}, t^*_{1-1} do not appear in the assembly anymore and the process continues with tiles of types t_{0-0}, t_{0-1}, t_{1-0} and t_{1-1}. After that, the process

repeats for the whole length of the support. Since the length of the support is at least N, the terminal assembly is a rectangle with exactly one tile of type t^*_{1-0} located at the left of the $(N+1)$st row. □



(a) Two full cycles of the assembly. (b) Zoom on the first cycle.

Figure 6.7: The assembly of the DBC System for number 12 to negative 15. The assembly goes on to the north until it reaches an obstacle. The red tile is the t^*_{0-1} that marks the left of row number $N + 1$.

6.4 The U-turn system

We now introduce a TAS which copies a row tile number to its left, as shown in Figure 6.8.

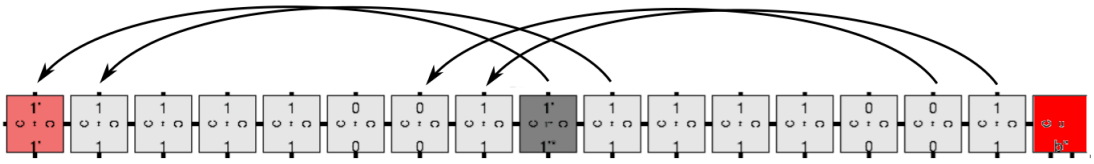


Figure 6.8: The goal of the U-turn system is to copy a row tile number to its left.

To do so, a U-shape rotation of each of its tiles will be performed (see Figure 6.12), which is why we gave the "U-turn" name to it. The seed is analogous to that of IBC and DBC.

Lemma 6.7 (U-turn system lemma)**.** *There exists a TAS* $\mathcal{S}_U = (\Sigma, T_U, \alpha_\sigma^U, str, \tau)$ *with the following properties. Assume a row tile number* N *is placed on a track of width at least* $2log(N)+1$ *and height at least* $log(N)$ *with the most significant bit placed on the coordinate* $(log(N)+1, log(N))$*. Then,* S *makes a copy of* N *to the left of its most significant bit. More precisely, if* N *was in positions* $[(x,y),...,(x+k,y)]$*, in the terminal assembly, there is a copy of* N *in positions* $[(x-k-1,y),\ldots,(x-1,y)]$*.*

The system is formally defined as follows.

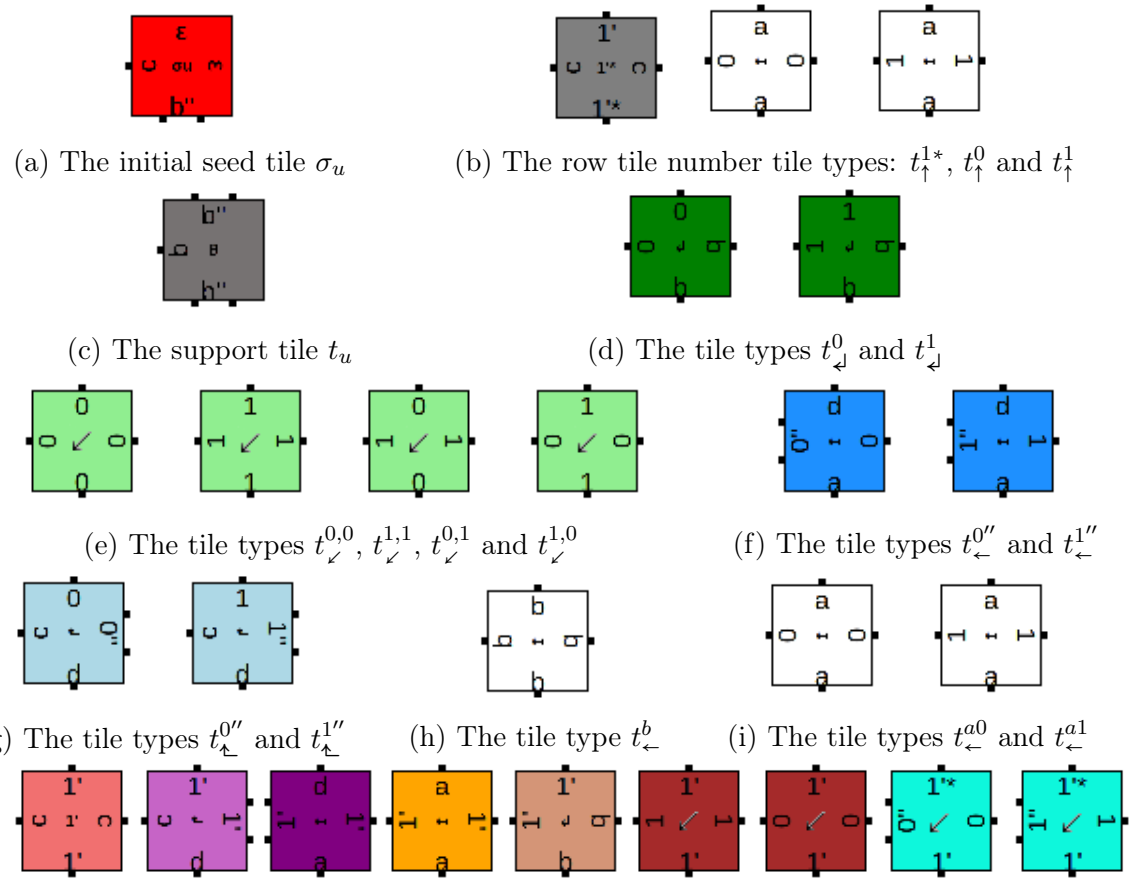
- $\Sigma = \{0, 0'', 1, 1'', 1', 1'^*, a, b, b'', c, d, \epsilon\}$
- $T_U = \{\sigma_u, t_u, t_{\leftarrow}^{0*}, t_{\leftarrow}^{1*}, t_{\leftarrow}^{0''}, t_{\leftarrow}^{1''}, t_{\leftarrow}^{a0}, t_{\leftarrow}^{a1}, t_{\leftarrow}^{b}, t_{\swarrow}^{0}, t_{\swarrow}^{1}, t_{\swarrow}^{0,0}, t_{\swarrow}^{0,1}, t_{\swarrow}^{1,0}, t_{\swarrow}^{1,1}, t_{\uparrow}^{1*}, t_{\uparrow}^{1}, t_{\uparrow}^{0}, t_{\Leftarrow}, t_{\nwarrow}^{0''}, t_{\nwarrow}^{1''}\}$.
- *The seed assembly* α_σ^U *is formed by a tile of type* $\sigma_u = (\epsilon, \epsilon, b'', c)$ *(where* ϵ *can be arbitrary labels) which is placed on the right of a row tile number, and at the north of the tile of type* σ_u*, a vertical column of support tiles of arbitrary size (with at least as many tiles as in the row tile number) with western label* b*.*
- $str : \Sigma \cup \{\epsilon\} \longrightarrow \{0,1,2\}$ *such that* $str(\epsilon) = 0$*. The values of the str function in* Σ *are* $str(0'') = str(1'') = str(b'') = 2$*, and* 1 *for the others.*
- $\tau = 2$

Proof. Let T_U be the set of tiles shown in Figure 6.9, its temperature is 2.

The seed and the block tiles in the U-turn system.

The U-turn system is a TAS that starts with a multiple seed such that the number of support tiles is at least the number of tiles of the row tile number. Let the tile $\sigma_u = (\epsilon, \epsilon, b'', c)$ bind to a row tile number N with $n = \lceil log_2 N \rceil$. We assume that the assembly is oriented as in Figure 6.10, where the initial row tile number is at the top right. From the south of σ_u, support tiles of type $t_u = (b'', \epsilon, b'', b)$ with northern and southern labels b'' whose strength are 2, are placed one after another. These tiles form a ribbon of tiles of type t_u that is perpendicular with respect to the seed. Once the number of support tiles becomes n, all these tiles together form the seed of the U-turn system, and the assembly begins.

After this step, the tiles below the seed start to grow and the assembly continues row by row. The overview of the assembly's process of the U-turn system is as below.



(a) The initial seed tile σ_u

(b) The row tile number tile types: $t^{1*}_{\uparrow}$, $t^{0}_{\uparrow}$ and $t^{1}_{\uparrow}$

(c) The support tile t_u

(d) The tile types $t^{0}_{\downarrow\!\!\!\downarrow}$ and $t^{1}_{\downarrow\!\!\!\downarrow}$

(e) The tile types $t^{0,0}_{\swarrow}$, $t^{1,1}_{\swarrow}$, $t^{0,1}_{\swarrow}$ and $t^{1,0}_{\swarrow}$

(f) The tile types $t^{0''}_{\leftarrow}$ and $t^{1''}_{\leftarrow}$

(g) The tile types $t^{0''}_{\nwarrow}$ and $t^{1''}_{\nwarrow}$

(h) The tile type $t^{b}_{\leftarrow}$

(i) The tile types $t^{a0}_{\leftarrow}$ and $t^{a1}_{\leftarrow}$

(j) The marked tile types for copying the most significant bit tile (which is of type $t^{1*}_{\uparrow}$)

Figure 6.9: The U-turn system and support tile types.

Overview of the assembly's process in the U-turn system.

The word "U-turn" refers to the process of the assembly in this system: each bit tile is copied along a U-shaped path to be copied to the left of the initial row tile number. More precisely, each tile a_k that is placed at the k-th bit of the row tile number, is copied k times to the south, then it is copied n times to the west, and at the end k times to the north. Furthermore, the assembly progresses row by row under the seed. Indeed, the last tile of each row has strength 2 at the west and increments the number of row tiles. Thus, the next tile is placed at the $(n+k)$-th column (starting from the rightmost column of Figure 6.10). This tile makes a base, and using the support of the last tiles of each row, a vertical column is constructed that sends the value of a_k to the north.

The first stage of the assembly in the U-turn system.

First, we explain the tiles of the very first row under the seed. For an illustration during this stage see Figure 6.11. This row is made of the tiles that transfer the value of the rightmost tile i.e. the least significant bit a_1 of the row tile number, to the column $n+1$. At first, a tile of type $t^{0}_{\downarrow\!\!\!\downarrow} = (0, b, b, 0)$ or $t^{1}_{\downarrow\!\!\!\downarrow} = (1, b, b, 1)$ (depending on the value of the least significant bit, for short we use $t_{\downarrow\!\!\!\downarrow}$) appears in the assembly. The tiles of type $t_{\downarrow\!\!\!\downarrow}$ transfer the value of the north label to the west label. These tiles are colored in dark green in Figure 6.10.

Then, tiles of type $t^{1,1}_{\swarrow} = (1, 1, 1, 1)$, $t^{1,0}_{\swarrow} = (1, 0, 1, 0)$, $t^{0,1}_{\swarrow} = (0, 1, 0, 1)$ and $t^{0,0}_{\swarrow} = (0, 0, 0, 0)$ (for short $t_{\swarrow}$, light green in Figure 6.10), that copy the north label to the south label, and the east label to the west label, appear under the row tile number N, at the west of the

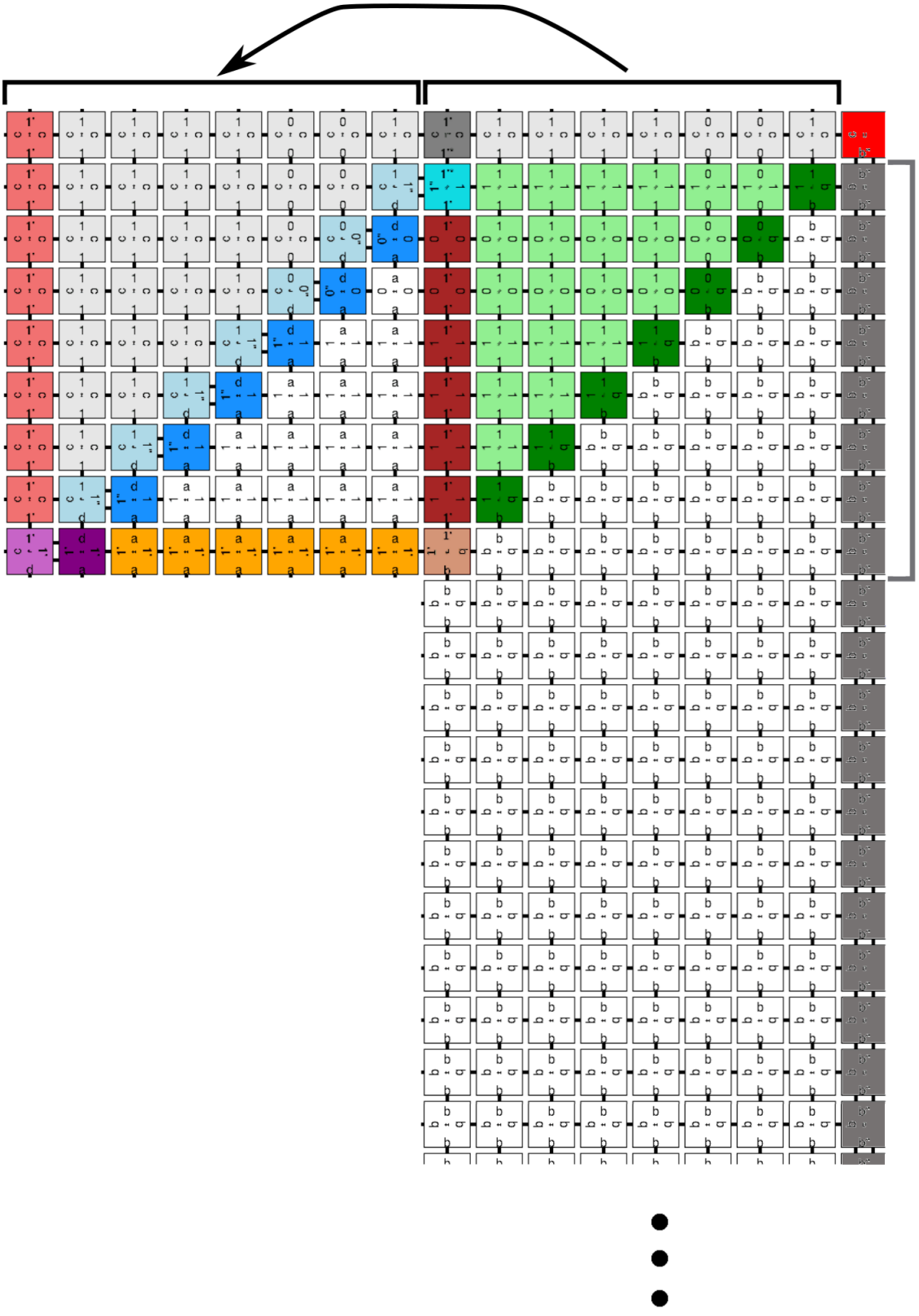


Figure 6.10: Copying 11111001 in the U-turn system. The gray bracket on the right shows the minimum number of support tiles that are necessary for this assembly.

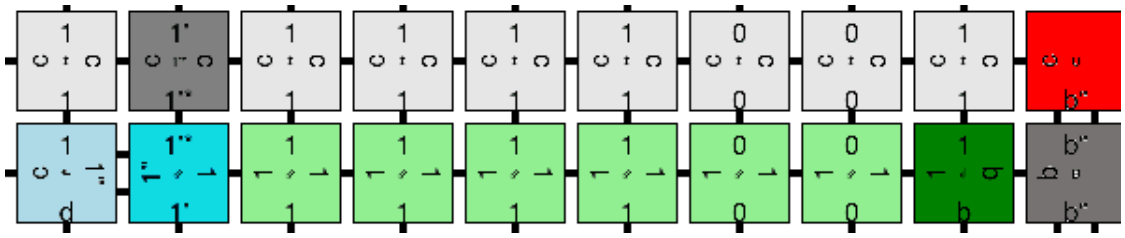


Figure 6.11: In the first stage of the assembly of the U-turn system, the value of the least significant bit is transferred n times to the left (here $n = 8$).

tile of type $t_{\downarrow}$ (dark green in Figure 6.10) except the leftmost one, which corresponds to the most significant bit. Notice that in the U-turn system, all the tile types that transfer the value of the most significant bit tile are distinct from the other tiles.

The tile below the most significant bit is a special tile of type $t^{1*}_{\leftarrow} = (1'^*, 1, 1', 1'')$ or $t^{0*}_{\leftarrow} = (1'^*, 0, 1', 0'')$ (cyan in Figure 6.10) that copies the north label to the south label and the east label to the west label with strength 2. Thus, this tile of type $t^{1*}_{\leftarrow}$ or $t^{0*}_{\leftarrow}$ allows to open a new column. The tile after it is of type $t^{0''}_{\nwarrow} = (0'', d, c, 0)$ or $t^{1''}_{\nwarrow} = (1'', d, c, 1)$ (light blue in Figure 6.10) with strength 2 at the east label and brings its value to the north. Now, the first copied tile, which corresponds to the least significant bit tile, appears at the left of the most significant bit tile of N.

Note that the tile of type $t_{\downarrow}$ also copies the east label, i.e. block label b from the support tiles t_u, to the south label. Thus, at the south of the tile of type $t_{\downarrow}$, the tiles of types $t^{b}_{\leftarrow} = (b, b, b, b)$ (named block tiles) appear vertically with the support of t_u.

The overall transfer of the value in the U-turn assembly.

Now, we explain the general process of the assembly for transferring the value of the k-th least significant bit of the row tile number. In Figure 6.12 this stage is marked with yellow-bordered rectangles on the left side of the assembly. In this part, each new tile can be placed if and only if their northern and eastern labels correspond to the previous tiles.

Assume that we are in the k-th stage i.e. the value of the k-th least significant bit is being transferred.

In the previous stages before k, the block tiles of types $t^{b}_{\leftarrow}$ grow vertically from the south of the tile of type $t_{\downarrow}$. Thus, the tiles of type $t^{b}_{\leftarrow}$ are placed between the support tiles of type t_u and $t_{\downarrow}$ in each row and copy the block label b from the east to the west. Therefore, the tile $t_{\downarrow}$ appears in the k-th row and the k-th column with help of the tiles of type $t_{\swarrow}$ in the north and $t^{b}_{\leftarrow}$ at its east. Then, the tile that is placed at the k-th bit of the row number is copied $k-1$ times to the south before it meets the block tile, the place where the tile of type $t_{\downarrow}$ (that transfers the north label to the west label) appears.

Then, the value of the k-th bit tile is copied one time to the south, and n times to the west. Before arriving at the $(k+n)$-th column, the value passes through the n-th column, i.e. the column of the most significant bit tiles.

As mentioned before, the path that the most significant bit goes through is created with marked tiles to keep it distinguished from the other bit tiles. From the column of the most significant tile types, the tiles of type $t^{a}_{\leftarrow}$ transfer their eastern value to the western value until arriving to the $(n+k)$-th column at the left. Note that, at the $(n+k-1)$-th bit towards the left, a tile of type $t^{1''}_{\leftarrow} = (d, 0, a, 0'')$ or $t^{0''}_{\leftarrow} = (d, 1, a, 1'')$ (dark blue in Figure 6.10) open a new column that starts with $t^{0''}_{\nwarrow} = (0'', d, c, 0)$ or $t^{1''}_{\nwarrow} = (1'', d, c, 1)$ (light blue in Figure 6.10) that transfers the value from the west to the north. Observe in Figure 6.10 that these tiles form a light blue diagonal bisector between the column of the

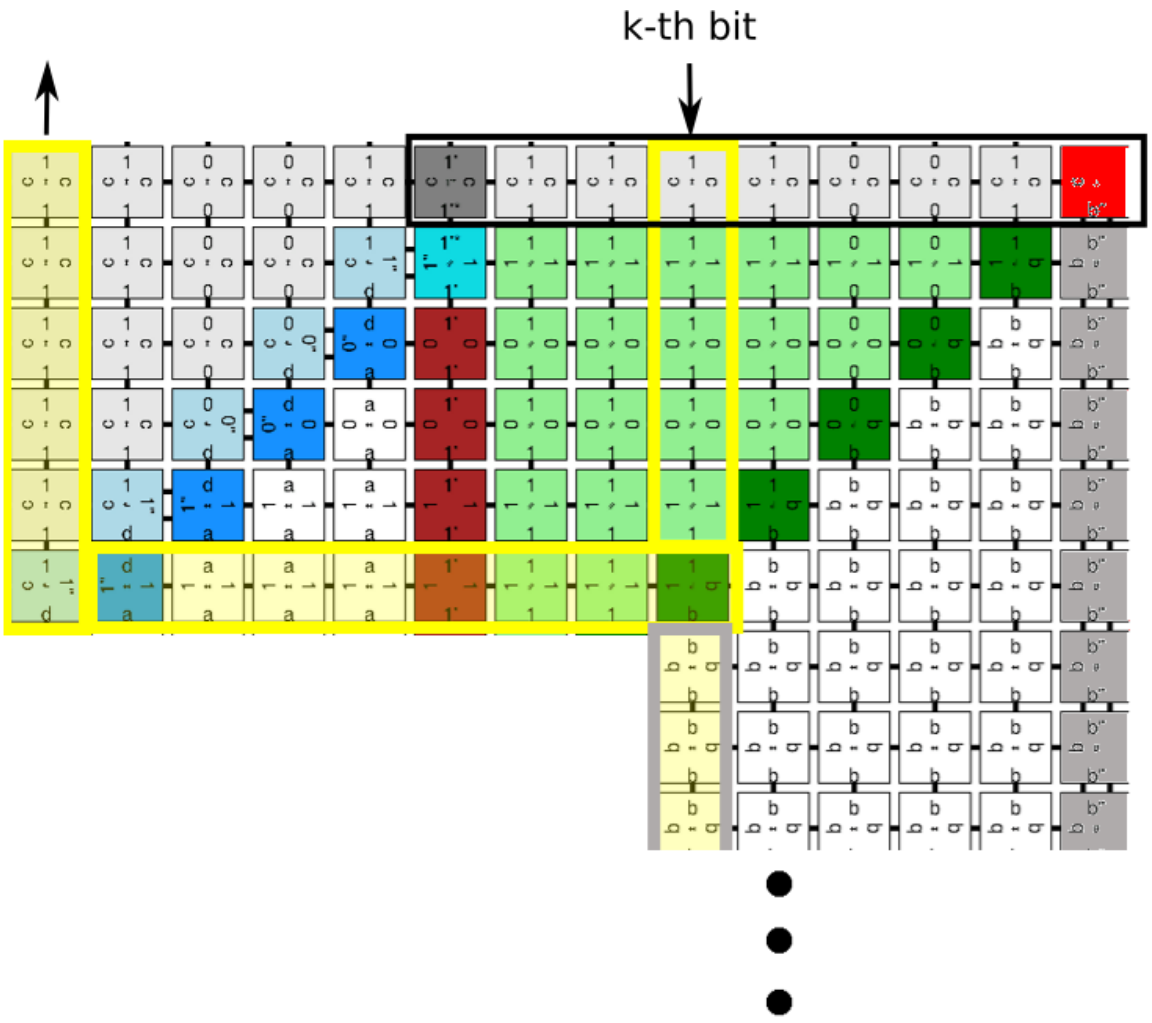


Figure 6.12: The k-th stage of the assembly in the U-turn system is shown by yellow-filled rectangles. The value of the k-th significant bit is copied down by $k-1$ rows during the previous stages. The k-th stage copies the value one time to the south and n times to the left and finally k times to the top. Here, $k = 5$ and $n = 8$. In addition, in the k-th stage, the tiles of type $t^b_{\leftarrow}$ in the gray rectangle appear below the tile of type $t_{\downarrow}$, and they will be the supports for the $(k+1)$-th stage. The seed is highlighted in the black rectangle.

most significant bit below it, and the copied row tile number of N. After these tiles are placed, the value of each tile is copied to the north using $t_{\uparrow}$-type tiles. Note that the tile types of the $(n+k)$-th column are placed by matching with their southern and eastern tiles.

This process continues until the value of each tile bit is placed in a new tile bit in the same row as N but with a shift of n columns. In the terminal assembly, there is a copy of of N (which was in positions $[(x,y),...,(x{+}k,y)]$) in positions $[(x{-}k{-}1,y),\ldots,(x{-}1,y)]$. $\square$

6.5 The middle finding system

Now we are ready to present the main result of this chapter. The following lemma uses the IBC, DBC and U-turn systems to present a SFTAM TAS in order to find the middle of the height of R. See Figure 6.13.

Lemma 6.8 (Middle finding system lemma)**.** *Let R be an explicitly bounded track of width at least $3\log(N)$, where N is the height of R, and the two rows of R having tiles form the sets Start in the south side and Finish in the north side, respectively. There is a TAS $S_{1/2} = (\Sigma, T_{1/2}, \alpha_\sigma^M, str, \tau)$ such that for all assemblies with a seed located at the start, there is a tile t_m which appears only at coordinate $(x, \lfloor \frac{N}{2} \rfloor)$ with no other tiles to its left. Formally the system is:*

- $\Sigma = \Sigma_{IBC1} \cup \Sigma_{IBC2} \cup \Sigma_U \cup \Sigma_{DBC} \cup \{org, c\}$ *where the labels in Σ are the same as in each system, except when a modification is explicitly mentioned. However, any two labels from two distinct systems are distinct from each other.*
- $T_{1/2} = T_{IBC1} \cup T_{IBC2} \cup T_U \cup T_{DBC} \cup \{t_v, t_{copy}\}$
- *The seed assembly α_σ^M is composed of a tile of type σ_+ from the IBC System, together with the tiles of Start and Finish and with a column of an arbitrary number of* support tiles *(the ones from the IBC System) placed on the north side of σ_+, all of whose west labels are* 1 *(the other labels can be arbitrary), going up to the Finish row.*
- $str : \Sigma \cup \{\epsilon\} \longrightarrow \{0, 1, 2\}$ *such that $str(\epsilon) = 0$ and $str(org) = str(c) = 1$. The values of the str function in Σ are the same as each system.*
- $\tau = 2$

Proof. Let R be an explicitly bounded track as described in the statement, and $N = 2^n + k$ with $k < 2^n$ be the height of R. We assume that the specially marked horizontal sides of R form the sets *Start* at the bottom and the other, $Finish$, at the top, as in Figure 6.13.

We use the IBC, DBC and U-turn systems from the previous sections to define our middle finding SFTAM TAS that finds $\frac{N}{2} = 2^{(n-1)} + \frac{k}{2}$. There are two copies of the IBC, named $IBC1$ and $IBC2$. To avoid any confusion regarding tile types of IBC, DBC and U-turn systems each label is marked with its system name.

For example, we use 1_{IBC} for the IBC system, and 1_{DBC} for the DBC system. However, whenever there is no ambiguity, we only use label 1 for simplicity. See an overview of the assembly in Figure 6.13 to follow the structure of the proof. Now we describe the steps for the middle finding system.

0. Growing a column of tiles of type t_{++} until "Finish".

The assembly starts to grow when σ_+ from the $IBC1$ System is placed besides the starting tiles. The tiles of type t_{++} pave the way until becoming blocked by "Finish" tiles, creating a column of tiles. These tiles are the support for the $IBC1$ system.

1. Using the IBC System $IBC1$ to find the height $N = 2^n + k$ of R.

Recall from Lemma 6.5 of IBC systems, that each row tile number represents the height of the assembly along the y axis. At the end, the assembly is blocked at the endpoint tiles of R in the row tile number N.

Here we modify the labels of the most significant bit tiles. We set $t_S = (1_S, 0, 1, 1)$ instead of $t_S = (1_S, 0, 1, \epsilon)$, i.e the last label is changed to 1. This change will be used for starting the growth of the second IBC system. In addition, instead of $t_O = (1_s, over, \epsilon, \epsilon)$ we

have $t_O = (1_s, over, org, org)$ ("org" stands for "orange"). The reason for this modification is that it will be useful in the third step, where we copy the value of k.

Here, because of this modification, two types of tiles will be able to be attached to the assembly. These tiles will be useful for the later stages of the assembly. Firstly, each time that in the IBC system, a new most significant bit tile is added, a vertical tile bond grows in its south by $t_v = (org, 1, org, 1)$, where org is the new southern label of t_O, and the label 1 is 1_{copy}. (These tiles will be used in the copy system.) Secondly, when t_v tiles collide with previous t_v tiles, a double column of tiles of type t_b appears. (These tiles will be used as block tiles in the U-turn system.)

2. Returning to row number 2^n using the second IBC system $IBC2$, which then outputs the value of k

Afterwards, at the collision of $IBC1$ with the finish block tiles, $IBC2$ appears next to the most significant bit tile of $IBC1$. This is where our modification of $IBC1$ becomes useful. In fact, changing the fourth label of the most significant bit tile of the $IBC1$ system to 1, enables this new IBC system $IBC2$ to start growing, using the most significant column of $IBC1$ as its support.

Note that due to the SFTAM rotating tiles, the two IBC systems are able to grow in opposite directions. Thus, the assembly returns towards the row tile number of 2^n using the $IBC2$ system.

consequently, the value of the last row tile number of $IBC2$ is k.

3. Copying k until 2^{n-1} (the middle of 2^n).

Now, the topmost column of tiles of type t_v from the first step will act as the support for the third step. Thanks to the vertical tile bond that is located below the most significant bit tile of the 2^n row tile number, the k-row tile number is copied 2^{n-1} times until the 2^{n-1}-row tile number of $IBC1$ (this is half of 2^n). To this aim, we use tiles of types $t_{copy} = (x, copy, x, copy)$ and $t^*_{copy} = (x, copy, x, 1_{DBC})$ (for $x \in \{0, 1\}$) such that x is the label of the IBC system, and t^*_{copy} is used for copying the most significant bit tiles of the IBC system.

Since the column of tiles of type t_v is the support of this step, the copy process stops after 2^{n-1} copies.

4. Halving k by eliminating its least significant bit.

Now, the copied row tile number with value k is halved by getting rid of its least significant bit. The block tiles that already appeared when two vertical tile bonds of type t_v met each other, discard the least significant bit tile of the k-row tile number and copy the $\frac{k}{2}$-row tile number to the left of the first block tiles. The tiles here are a tile of type $t_{b'} = (org, c, \epsilon, u)$ such that u is a label with strength 2, and a seed σ_u of the U-turn system that binds to it. These are block tiles. Also, from the south of σ_u, there are tiles of type t_u, the support tiles of the U-turn system. They are shown in black in Figure 6.13.

5. Shifting $\frac{k}{2}$ to the left by a U-turn system.

The column of tiles of type t_u from the previous step form the support for this step. The binding of the block tiles to the $\frac{k}{2}$-row tile number form a U-turn block seed and the $\frac{k}{2}$-row tile number shifts to its left. Remember that the U-turn system needs space for the $\frac{k}{2}$ tiles in the south of the U-turn seed; this condition is true at the row number 2^{n-1} along the y axis, since $\frac{k}{2} < 2^{n-1}$.

6. Going up by $\frac{k}{2}$ using the DBC system.

After shifting $\frac{k}{2}$ to the left at the row $\frac{2^n}{2}$, a DBC system starts from $\frac{k}{2}$ along the side of the Copy system. Note that here the label 1_{DBC} of t^*_{copy} (from the Copy system) plays the role of the support tile type in this DBC system. Moreover, the first row of the DBC system i.e. the seed, is the copied $\frac{k}{2}$ row tile number from the U-turn system. We add two final modifications to the DBC. We change the tile type t^*_{0-1} to $t_m = (1', 0, 1', --)$ by changing the fourth label.

After passing the zero with the DBC system, a tile of type t_m appears in row $\frac{N}{2} = 2^{n-1} + \frac{k}{2}$. The process of finding the middle finishes here by the appearance of t_m in the assembly.

At the end, note that a tile of type t_m only appears once, because the support of the DBC system is the left side of the copy system. Moreover, in the middle finding system on an explicitly bounded track, we have a width of at least $3\log(N)$, where N is its height so that IBC1, IBC2 and DBC with IBC Support systems are able to grow without any space restriction. For an outline of the middle finding system, see Figure 6.13, where t_m is in red on the left. □

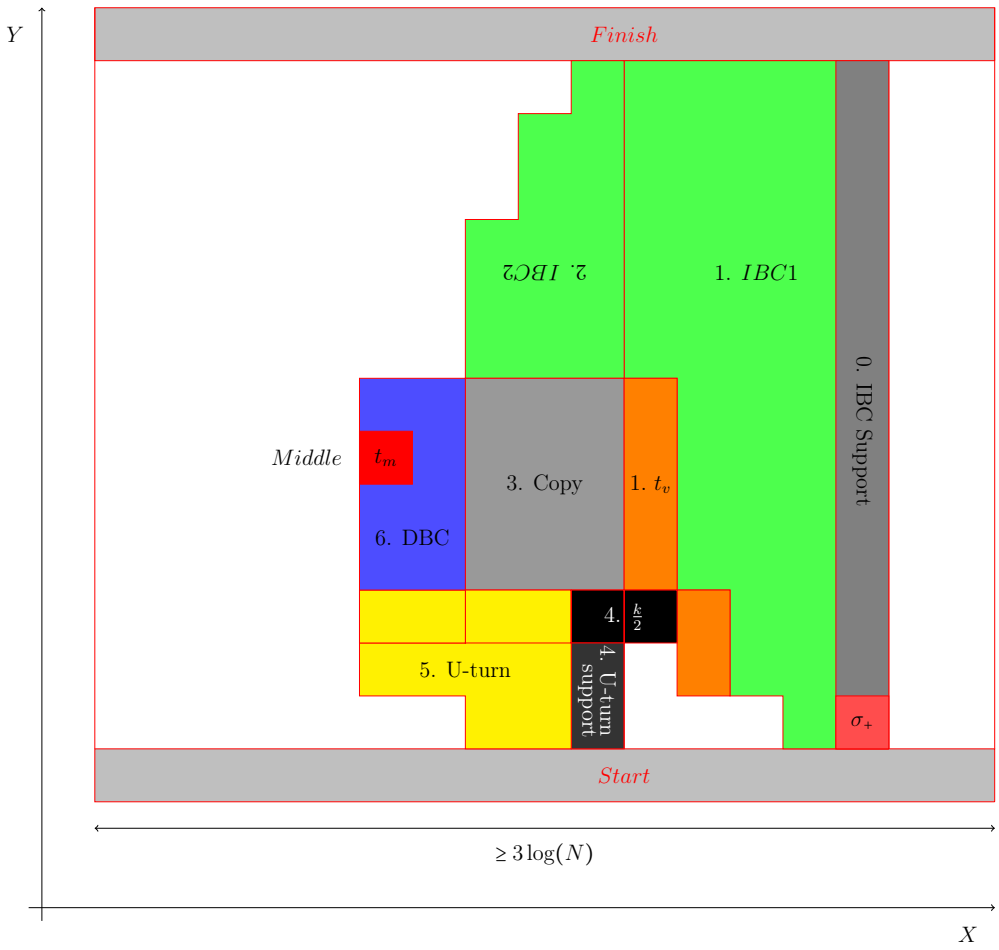


Figure 6.13: The steps of the middle finding system process. The tile t_m is the left red one.

6.6 Concluding remarks

We have proposed the middle finding system, a TAS that works both in the aTAM in $\mathbb{Z}^2$ and (via Lemma 5.13) in the SFTAM on polycubes, to find the middle of a track. Note that both the increasing and decreasing binary counters are essential here, as without them it would not be possible to determine the length of the track.

We believe that this TAS may be useful for future works in other contexts.

The middle finding system for polycubes paves the way to identify the genus of some 3D surfaces in Chapter 7.

Chapter 7

Detecting the genus of order-1 cuboids using the SFTAM

This chapter deals with detecting the genus of a family of surfaces of 3D objects called *cuboids*, which are special types of polycubes. Polycubes can form complex surfaces, and their genus can be high. We focus on a simple family of polycubes that can have genus 0 or genus 1, called *order-1 cuboids*. In this chapter, we will suppose that the SFTAM self-assembly is performed on the surface of an order-1 cuboid C. We design a SFTAM tile assembly system which is able to distinguish the order-1 cuboids with genus 1 from the others using its terminal assemblies. The definitions and lemmas from Chapter 6 will be needed for this task. A brief description of our main result is sketched as follows.

There is a SFTAM tile self-assembly system $\mathcal{S}_\mathcal{G}$ and a subset of tile-types $Y \subseteq T$ such that for an order-1 cuboid C, if $\mathcal{S}_\mathcal{G}$ assembles on C starting from a seed which is placed in a proper way, the following holds:

- if C has genus 1, every terminal assembly of $\mathcal{S}$ on C contains at least one tile of Y, and
- if C has genus 0, then no tile of Y appears in any producible assembly of $\mathcal{S}$ on C.

In other words, the genus of C can be determined using the assemblies of $\mathcal{S}_\mathcal{G}$ on C.

The goal of this study is to show that performing self-assembly on surfaces of higher genus can be helpful. We also demonstrate some techniques which may prove useful in characterizing the topological properties of a wide range of surfaces.

In Chapter 4, we obtained a similar classification for four types of flat surfaces. As these surfaces are more regular than cuboids, our task was much easier. Indeed, the position of the seed was not important, as the assemblies were invariant under translations. In the case of cuboids however, this is not the case and we have to come up with a more complicated strategy to distinguish between genus 0 and genus 1 cuboids.

As noted in Remark 3.32, tile self-assembly is asynchronous and nondeterministic. Thus, one of the main difficulties we need to overcome is to control the assembly. For that, in order to prevent such conflicts, we will have several successive phases in our assembly.

Remark 7.1. *For simplicity of presentation, we do only specify labels as l and not as the complement $\bar{l}$. However, if a TAS has a label l on some west sides and on some east sides of tiles, we implicitly assume that the western labels are actually l and the eastern labels are the complement $\bar{l}$. Similarly, if a label l is present both on the north/south*

of some tiles and on the east/west of some tiles, we implicitly assume that the label for the north/south is l_V and the label for east/west sides is l_H, to prevent a rotation and unwanted attachment.

We start with defining the cuboids in Section 7.1. We then formally state the main theorem of this chapter, Theorem 7.6, in Section 7.2. We give an overview of the framework in Section 7.3. We describe the TAS $\mathcal{S}_\mathcal{G}$ in Section 7.4. We then give a closer intuitive idea of the proof in Section 7.5. We then describe in more detail the assemblies of $\mathcal{S}_\mathcal{G}$ in Section 7.6, and we finalize the proof of Theorem 7.6 in Section 7.7. We conclude in Section 7.8.

7.1 The order-1 cuboids

In this section, we introduce the structures that we work on: *order-1 cuboids*, which are special types of polycubes. We work in 3-dimensional space, on the integer lattice $\mathbb{Z}^3$. We start with some definitions.

Definition 7.2 (Order-0 cuboid)**.** *An* order-0 cuboid $C = (s_C, x_C, y_C, z_C)$ *is a 3D structure, where $s_C = (s_x, s_y, s_z) \in \mathbb{Z}^3$ is the point of C with smallest coordinates and x_C, y_C, z_C are integers representing the length, width and height of C. It contains all points (x, y, z) of $\mathbb{Z}^3$ such that $s_x \leq x \leq s_x + x_C$, $s_y \leq y \leq s_y + y_C$ and $s_z \leq z \leq s_z + z_C$. We denote the set of all cuboids by O_0.*

Note that we work on the boundary surface of cuboids.

We are interested in 3D structures that are more complicated than order-0 cuboids, in particular 3D structures that can have *tunnels*, that is, "holes". To this aim, we introduce a family of polycubes called *order-1 cuboids*. A cuboid is *connected* if one can reach any point of its surface from any other point by a connected path.

Definition 7.3 (Order-1 cuboid)**.** *An* order-1 cuboid C_1 *is a connected cuboid that is the difference between two order-0 cuboids. Given $C_0 = (s_{C_0}, x_{C_0}, y_{C_0}, z_{C_0})$ and $C'_0 = (s_{C'_0}, x_{C'_0}, y_{C'_0}, z_{C'_0})$ in O_0, $C_1 = C_0 \setminus C'_0$ is an order-1 cuboid if there is a $i \in \{x, y, z\}$ such that $i_{C_0} \leq i_{C'_0}$. We note O_1 the set of all order-1 cuboids.*

Note that an order-0 cuboid is a classic "cuboid", i.e. it has six rectangular faces. An order-1 cuboid can have an asymmetric surface, including a hole or a concavity. Moreover, the condition that there is a $i \in \{x, y, z\}$ such that $i_{C_0} \leq i_{C'_0}$ ensures that there is no "hole" in C_1 that would be unreachable from its surface.

The genus of an order-1 cuboid is at most 1. Note that the set of order-0 cuboids is a subset of the set of order-1 cuboids, that is, $O_0 \subseteq O_1$.

Definition 7.4. *An order-1 cuboid $C_1 = C_0 \setminus C'_0$ can be of three different types, depending on how C_0 and C'_0 interact: (i) C_0 and C'_0 have no intersection, and C_1 is an order-0 cuboid, (ii) C'_0 cuts a hole in C_0, and C_1 is called an order-1 cuboid with a* tunnel*; and (iii) C_0 and C'_0 intersect but C'_0 does not cut a hole in C_0. If the cut is in the inner face of C_0, C_1 is an order-1 cuboid with a* pit *and if the cut is in the side of a face of C_0, C_1 is an order-1 cuboid with a* concavity*.*

An order-1 cuboid with a tunnel has genus 1. In both cases of order-1 cuboid with a pit or with a concavity, the genus is 0. See Figure 7.1 for illustrations of the four types of order-1 cuboids.

Definition 7.5. *We denote by O_1^* the set of order-1 cuboids that are not order-0 cuboids, that is, the ones from items (ii) and (iii) above.*

The set of cuboids of O_1^* with a pit are denoted by O_1^p, the ones with a tunnel, by O_1^t, and the ones with a concavity, by O_1^c.

7.2 Statement of the main theorem

We now state our main result.

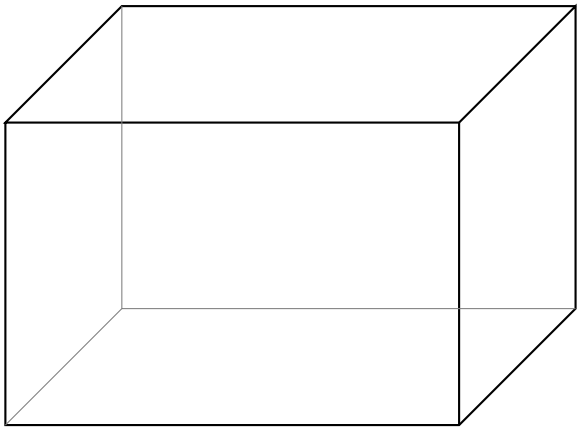
Theorem 7.6. *There is a SFTAM tile self-assembly system $\mathcal{S}_\mathcal{G} = (\Sigma, T, \sigma, str, \tau)$ and a subset of tile-types $Y = \{t_{reg}, t_{mfs}, t_{ibc1}, t_{ibc2}\} \subseteq T$ such that for any order-1 cuboid $C = C_0 \setminus C'_0$ with the dimensions at least 10 for C'_0, if $\mathcal{S}_\mathcal{G}$ assembles on C starting from a seed which is placed in a normal placement, the following holds:*

- *if C has genus 1, every terminal assembly of $\mathcal{S}$ on C contains at least one tile of Y, and*
- *if C has genus 0, then no tile of Y appears in any producible assembly of $\mathcal{S}$ on C.*

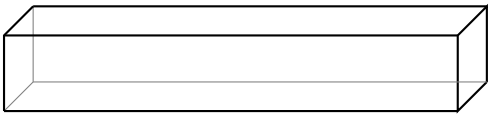
There is a crucial point to consider: the system presented here works for the case where the assemblies' seed is placed on a *normal placement* p of an order-1 cuboid, i.e. the place p is "far enough" from the boundaries of the face of the surface where the placement belongs to, so that there is enough space for our assemblies. The notion of a normal placement is formalized in the following definition.

Definition 7.7 (Normal placement)**.** *Let C be an order-1 cuboid such that (x_1, y_1, z_1), (x_2, y_2, z_2), ... ,(x_n, y_n, z_n) are its vertices. A placement $p \in Pl(C)$ with position (x, y, z) is a* normal placement *of C if and only if for all $i \in \mathbb{N}$, two of the following inequalities hold: $| x_i - x | \geq 6\lceil \log(N) \rceil + 9$, $| y_i - y | \geq 6\lceil \log(N) \rceil + 9$ and $| z_i - z | \geq 6\lceil \log(N) \rceil + 9$, where N is the largest of the three dimensions of the cuboid. The set of all normal placements is denoted by $Pl_N(C)$.*

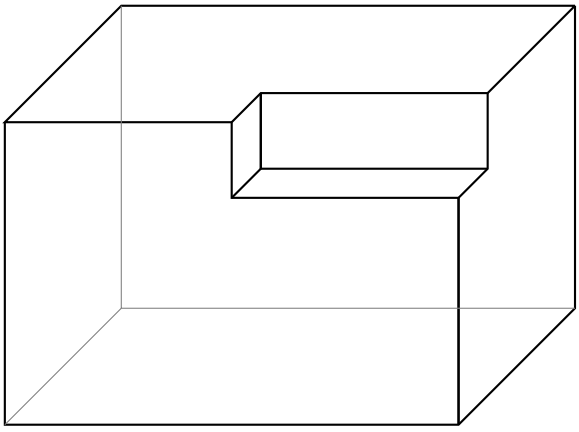
The simplest example to demonstrate the concept of normal placement is on a cuboid $C \in O_0$. In this case, normal placements consist of the cuboid's surface minus its "frame" i.e. the border of the cuboid's edges with a thick margin. Hence there are 6 disconnected



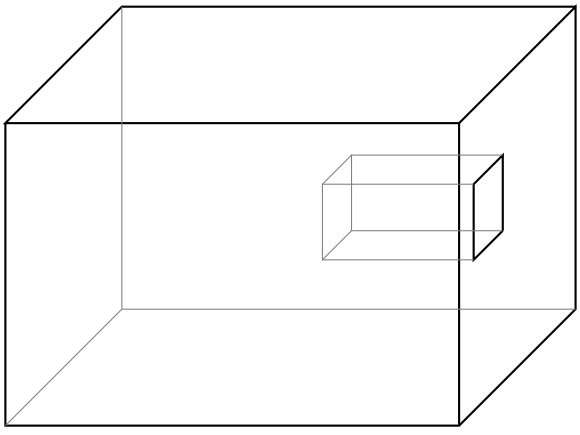
(a) Order-0 cuboid $C_0 \in O_0 \subset O_1$.



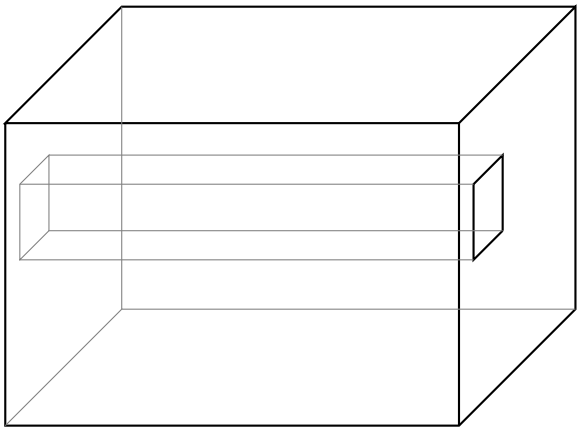
(b) Order-0 cuboid $C_0' \in O_0 \subset O_1$.



(c) Order-1 cuboid $C_c \in O_1^c$ with concavity, obtained by substracting a translated copy of C_0' from C_0.

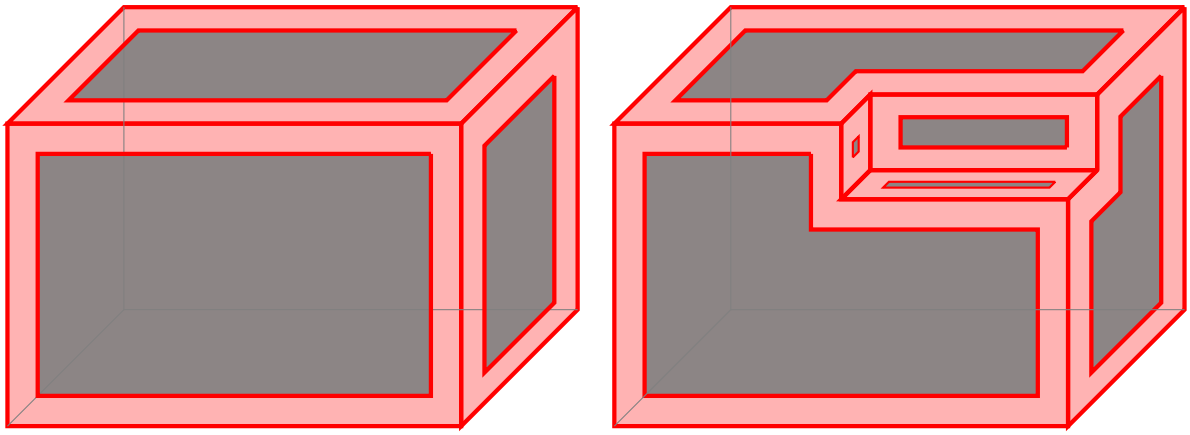


(d) Order-1 cuboid $C_p \in O_1^p$ with pit, obtained by substracting a translated copy of C_0' from C_0.



(e) Order-1 cuboid $C_t \in O_1^t$ with tunnel, obtained by substracting a translated copy of C_0' from C_0.

Figure 7.1: The four types of cuboids studied in this chapter: order-0 cuboids C_0 and C_0', and order-1 cuboids that are obtained by subtraction of (a translated copy of) C_0' from C_0.



(a) The normal placements of an order-0 cuboid C_0. The normal placements of C_0 contain 6 disconnected areas (here, 3 of them are visible from our point of view).

(b) The normal placements of an order-1 cuboid C_c with a concavity. The normal placements of C_c belong to 9 disconnected areas (here, 6 of them are visible from our point of view).

Figure 7.2: Illustration of the normal placements on two order-1 cuboids from Definition 7.7. Intuitively, these are the placements p that are sufficiently far from every edge belonging to the same face as p. The normal placements are the placements in the gray area, and the placements that are not normal are in the red areas.

areas on the faces where the normal placements are. The normal placements on order-1 cuboids can be described similarly. It should be noted that in this case there can be more than 6 disconnected areas: there are up to 9 disconnected areas with normal placements for $C \in C_1^c$, 10 for $C \in O_1^t$ and 11 for $C \in O_1^p$, depending on the new faces that are generated by the nature of the order-1 cuboid's construction. For example in Figure 7.2, the normal placements are shown by the areas colored in gray.

Remark 7.8. *Note that when the smallest dimension is large enough with respect to the others, "almost all" the placements on order-1 cuboids are normal placements.*

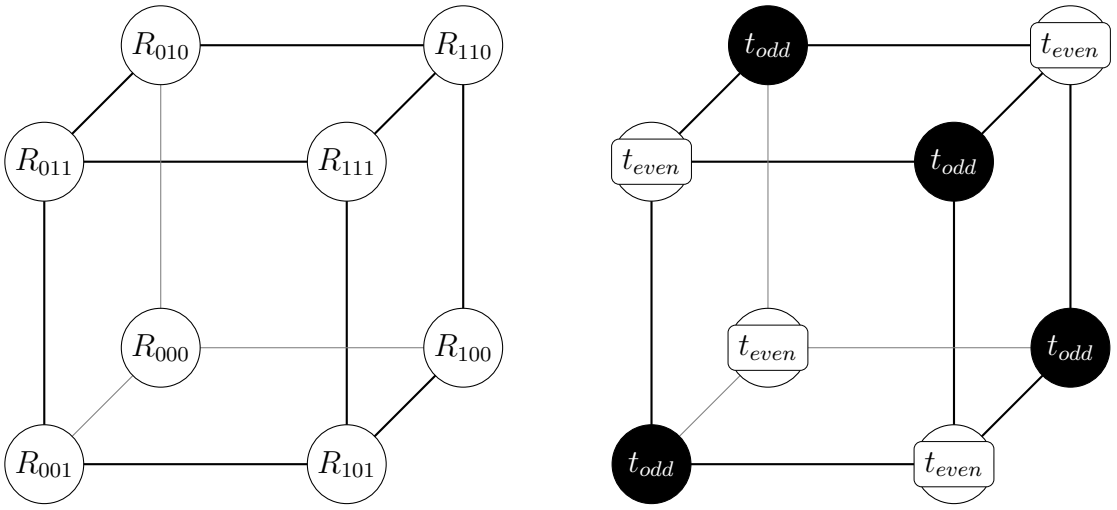
From now on, we assume that the seed of assemblies in $\mathcal{S}_\mathcal{G}$ is placed on a normal placement of an order-1 cuboid.

7.3 Our framework: region partition of order-1 cuboids

Let $C = C_0 \setminus C_0'$ be an order-1 cuboid. In order to detect a potential tunnel whose entrances are on parallel faces, the construction separates these faces. For this purpose we use three planes, one for each pair of parallel faces of C, located between them. Let P_X, P_Y, P_Z be three planes in this way: take $p \in Pl_N(C)$. The plane P_X is passing on p and is parallel to the plane formed by the Y-axis and Z-axis. The plane P_Y is parallel to the plane formed by the X-axis and Z-axis and is passing on p. The plane P_Z is parallel to the plane formed by the X-axis and Y-axis and contains the center of C_0. In Figure 7.9 the seed in yellow is in the point p and the plane P_X, P_Y and P_Z are framed respectively by the ribbons R_X (in red), R_Y (in green) and R_Z (in blue) on C. For $i \in \{X, Y\}$, R_i is the connected component of $\partial C \cap P_i$ that contains p. If R_X and R_Y intersect in one point, R_Z is the empty set. If they intersect in two points, $R_Z' = P_Z \cap \partial C$ and R_Z is the connected

component of R'_Z that has an intersection with R_Y. The difference $C \setminus \{R_X, R_Y, R_Z\}$ consists up to 8 connected components is called *regions*. They are noted by R_{XYZ} such that $X, Y, Z \in \{0, 1\}$ where 0 represents the left, down and back sides, and 1 represents the right, up and front sides. For example, R_{101} refers to the region at the right, down and front side of C. This way of partitioning C helps to define the graph G_C, the *region graph* of C:

Definition 7.9 (Region graph)**.** *Let $C = C_0 \setminus C'_0$ be an order-1 cuboid with p as a position on C, the planes P_X, P_Y two perpendicular planes passing on p, and P_Z a plane perpendicular to both planes passing through the middle of P_Y. Also, let $R_i = \partial C_0 \cap P_i$ for $i \in \{X, Y, Z\}$. There is a graph assigned to C named the* region graph *$G_C(p)$ whose vertices are the regions separated by R_X, R_Y and an edge is added between two regions if and only if they share P_X, P_Y or P_Z.*



(a) The region graph

(b) Tile coloring of the region graph, and filling up the regions via inner filling tiles of types t_{even} and t_{odd}.

Figure 7.3: The region graph G_C: $V(G_{S_G})$ are the distinct regions, and $E(G_{S_G})$ contains the edges between neighboring regions.

For an order-0 cuboid C, $G_C(p)$ is a bipartite graph and therefore it is 2-colorable. The region graph for an order-0 cuboid is presented in Figure 7.3.

If C is an order-1 cuboid with a tunnel, the number of disconnected regions can be less than 8 depending on the intersection of the tunnel with three planes. The *axis* of the tunnel is the direction orthogonal to its entrances. The three planes can intersect the tunnel in two ways: along the width of the tunnel when the plane is perpendicular to the axis of the tunnel, or along the length of the tunnel when the plane is parallel to the axis of the tunnel. Thus, a tunnel may have an intersection with up to three perpendicular planes, one along the width, and up to two other planes along the length. Based on this, three types of partitions into regions are possible and the possible numbers of regions are: 7 regions when one plane intersects along the width of the tunnel, 5 regions when one plane intersects along the length of the tunnel and one along the width (See Figure 7.21), and 1 region when three perpendicular planes intersect along the tunnel, one along the width and the others along the length.

7.4 Description of the TAS $\mathcal{S}_\mathcal{G}$

Before starting the details of $\mathcal{S}_\mathcal{G}$ assemblies, in this section we describe TAS $\mathcal{S}_\mathcal{G} = (\Sigma, T, \sigma, str, \tau)$. The tiles are presented in the following:

The tile types of the ribbon R_X:

- A seed tile $\sigma = (\epsilon, \epsilon, x', \epsilon)$
- $x' = (x', x''_e, x, x''_w)$ and $x = (x, x_e, x, x_w)$
- $x'_e = (x'_e, x'_e, x_e, x''_e)$ and $x'_w = (x'_w, x''_w, x_w, x'_w)$
- $x_e = (x_e, x_e, x_e, x_e)$ and $x_w = (x_w, x_w, x_w, x_w)$

These tiles are presented in Figure 7.4.

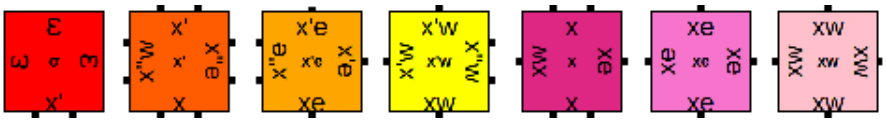


Figure 7.4: The tile types for R_X.

The tile types for R_Y:

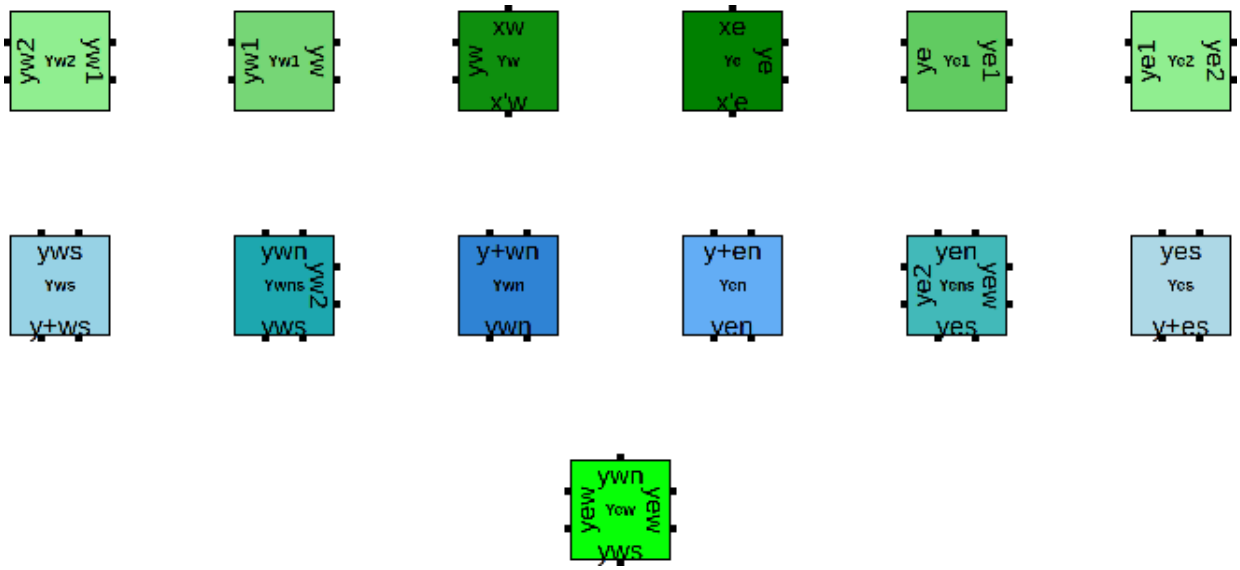
- The 13 starter tile types : $Y_{wn} = (y + wn, \epsilon, ywn, \epsilon)$, $Y_{wns} = (ywn, yw2, yws, \epsilon)$, $Y_{ws} = (yws, \epsilon, y + ws, \epsilon)$, $Y_{w2} = (\epsilon, yw1, \epsilon, yw2)$, $Y_{w1} = (\epsilon, yw, \epsilon, yw1)$, $Y_w = (xw, \epsilon, x'w, yw)$, $Y_e = (xe, ye, x'e, \epsilon)$, $Y_{e1} = (\epsilon, ye1, \epsilon, ye)$, $Y_{e2} = (\epsilon, ye2, \epsilon, ye1)$, $Y_{ens} = (yen, yew, yes, ye2)$, $Y_{en} = (y + en, \epsilon, yen, \epsilon)$, $Y_{es} = (yes, \epsilon, y + es, \epsilon)$, $Y_{ew} = (ywn, yew, yws, yew)$.
- The tile types of four middle finding systems are the similar to tile types presented in Lemma 6.8 with minor modifications on seeds and supports. The eastern systems MFS_{en}, MFS_{es} have respectively $\sigma^+_{en} = (over_{en}, {+}{+}_{en}, y{+}_{en}, \epsilon)$, $\sigma^+_{es} = (y{+}_{es}, {+}{+}_{es}, over_{es}, \epsilon)$ as seeds, and support tiles of type ${+}{+}_{en} = (1_{en}, {+}{+}_{en}, en, {+}{+}_{en})$ and ${+}{+}_{es} = (es, {+}{+}_{es}, 1_{es}, {+}{+}_{es})$. The western systems MFS_{wn}, MFS_{ws} have respectively seeds $\sigma^+_{wn} = (over_{wn}, \epsilon, y{+}_{en}, {+}{+}_{wn})$, $\sigma^+_{ws} = (y{+}_{ws}, \epsilon, over_{ws}, {+}{+}_{ws})$, and support tiles of type ${+}{+}_{wn} = (1_{wn}, {+}{+}_{wn}, wn, {+}{+}_{ewn})$ and ${+}{+}_{ws} = (ws, {+}{+}_{ws}, 1_{ws}, {+}{+}_{s})$.
  Apart from that, each system has also its own distinct other tile types of middle finding systems.

These tiles are presented in Figure 7.5.

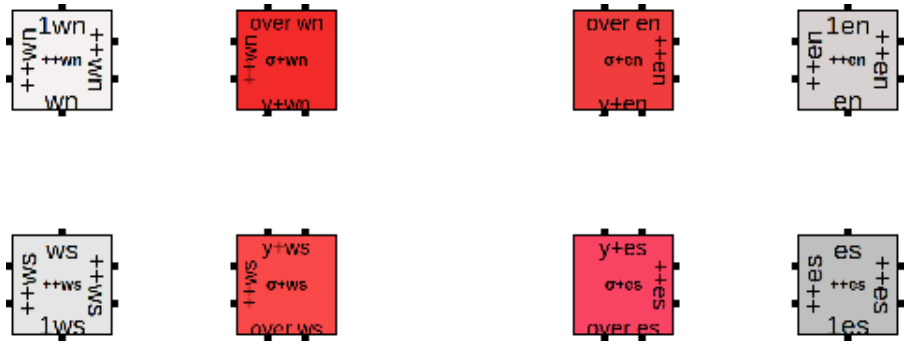
The tile types of R_Z:

- $t_{mr} = (1, 1, 0', z''r)$, $z_{1r} = (ze, z''r, z''r, zr)$, $z_{2r} = (z''r, \epsilon, zo, zo)$, $z_r = (ze, zr, r, zr)$, $z_{or} = (r, zo, zo, zo)$.
- $t_{ml} = (1, z''l, o', 1)$, $z_{1l} = (z''l, zl, ze, z''l)$, $z_{2l} = (zo, zo, zl'', \epsilon)$, $z_l = (l, zl, ze, zl)$, $z_{ol} = (zo, zo, l, zo)$.
  These tiles are presented in Figure 7.6.

The tile types of the inner filling: The 20 tile types of the four inner ribbons at the intersection of R_X and R_Z are presented in Figure 7.8, the tile types t_{even} and t_{odd} forming inner filling ribbons for regions and tile type $t_{reg} = (z_e, x_{od}, z_o, x_{ev})$ that appears only when the genus is 1 are:

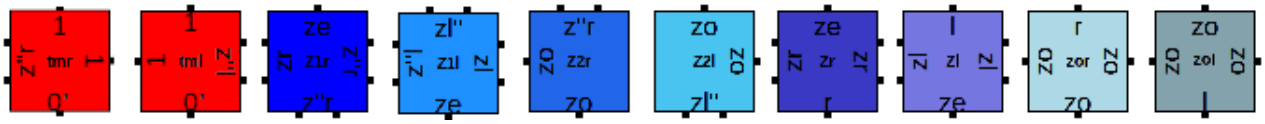


(a) The tile types which always appear in R_Y.



(b) The four distinct seeds and supports of MFS_{en}, MFS_{es}, MFS_{wn} and MFS_{ws}. Their other tiles are similar to the tile types presented in Lemma 6.8, but distinct for each one.

Figure 7.5: The tile types of R_Y.



(a) The tile types that always appear in R_Z ribbons.

Figure 7.6: The tile types for R_Z.

- $t_{ee} = (z_{e1}, x_{ev}, z_e, x_e)$, $t_{ee1} = (z_{e2}, \epsilon, z_{e1}, x_e)$, $t_{ee2} = (z_{e3}, \epsilon, z_{e2}, x_e)$, $t_{ee3} = (z_{e4}, \epsilon, z_{e3}, x_e)$, $t_{ee4} = (z_e, \epsilon, z_{e4}, x_e)$.
- $t_{eo} = (z_o, x_{od}, z_{o1}, x_e)$, $t_{eo1} = (z_{o1}, \epsilon, z_{o2}, x_e)$, $t_{eo2} = (z_{o2}, \epsilon, z_o, x_e)$.
- $t_{we} = (z_e, x_w, z_{e1}, x_{ev})$, $t_{we1} = (z_{e1}, x_w, z_{e2}, \epsilon)$, $t_{we2} = (z_{e2}, x_w, z_{e3}, \epsilon)$, $t_{we3} = (z_{e3}, x_w, z_{e4}, \epsilon)$, $t_{we4} = (z_{e4}, x_w, z_e, \epsilon)$.
- $t_{wo} = (z_{o1}, x_w, z_o, x_{od})$, $t_{wo1} = (z_{o2}, x_w, z_{o1}, \epsilon)$, $t_{wo2} = (z_o, x_w, z_{o2}, \epsilon)$.

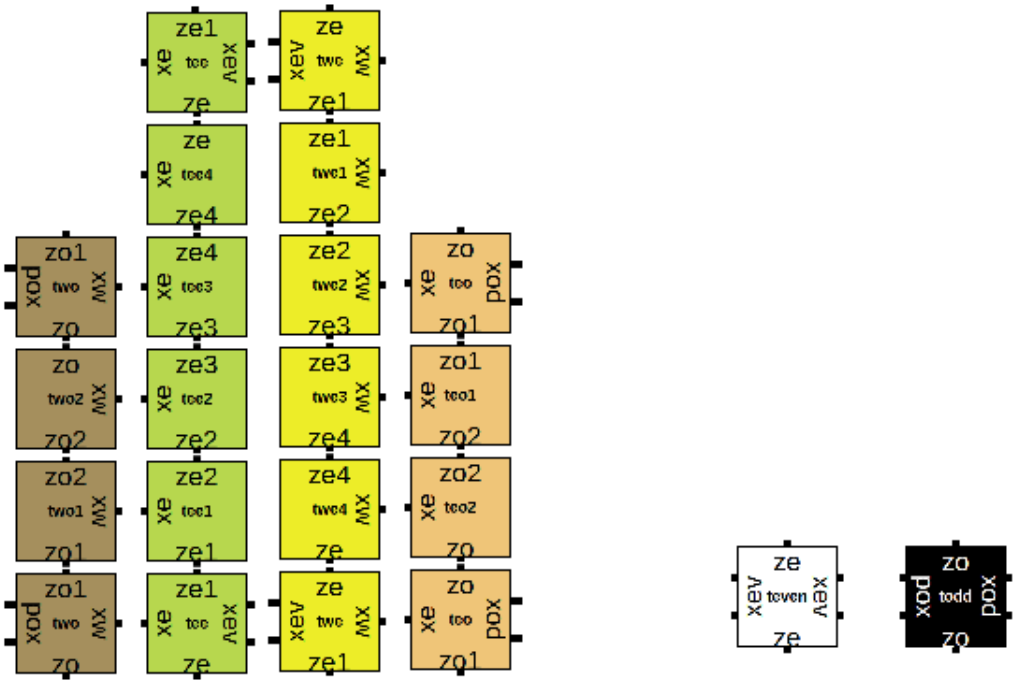
(a) The tile type t_{reg} that appears when there is a tunnel inside regions.

(b) The tile types t_{ibc1} and t_{ibc2}, which may only appear if C has genus 1, when the R_Y ribbons do not meet the R_X ribbons.

(c) The tile type t_{mfs}, which may only appear if C has genus 1, when the R_Z ribbons meet each other.

Figure 7.7: The tile types that show the presence of a tunnel in an order-1 cuboid.

– $t_{odd} = (z_o, x_{od}, z_o, x_{od})$ and $t_{even} = (z_e, x_{ev}, z_e, x_{ev})$.



(a) The tile types of four inner ribbons at the intersection of R_X and R_Z.

(b) The tile types of inner filling ribbons.

Figure 7.8: Tile types of inner filling of regions

The tile types which demonstrate a tunnel on the cuboid, and form the set Y:

– $t_{reg} = (z_e, \epsilon, z_o, \epsilon)$,
– $T_{ibc} = \{t_{ibc1}, t_{ibc2}\}$ such that $t_{ibc1} = (ws, \epsilon, y_w s, \epsilon)$ and $t_{ibc2} = (wn, \epsilon, ywn, \epsilon)$, and
– $t_{mfs} = (z_e, z_o, z_e, z_o)$.
These tiles are showed in Figure 7.7.

The temperature of the system $\mathcal{S}_\mathcal{G}$ is 2. The alphabet of the system is the unions of all labels on the side of tiles and the strength of each label is equal to the number of black sticks on the sides. The label ϵ does not necessary needs to appear on the sides of tiles.

7.5 Overview of the assemblies of $\mathcal{S}_\mathcal{G}$ on O_1 and proof ideas

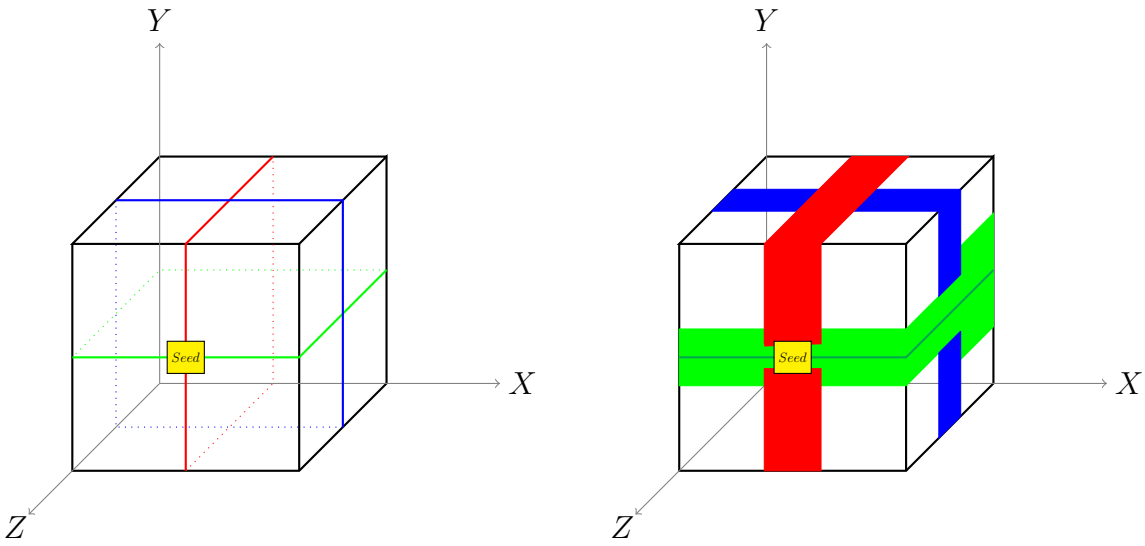


Figure 7.9: The skeleton of a terminal assembly of $\mathcal{S}_\mathcal{G}$ on an order-0 cuboid starting from a seed (in yellow) in a normal placement. On the left, the traces of the ribbons R_X (in red), R_Y (in green) and R_Z (in blue). On the right, the shape of the skeleton.

Let C be an order-1 cuboid. An assembly of $\mathcal{S}_\mathcal{G}$ starts from a seed in an arbitrary normal placement on C. In the TAS $\mathcal{S}_\mathcal{G}$, the seed acts like a compass for the assemblies. Without loss of generality, we assume that the side on which the seed is located is the face parallel to the XY-plane and intersects the Z axis, and the north label of the seed's tile points towards the Y axis. In order to control the assemblies' growth of S_G, it is structured into different successive phases, that cannot overlap. It consists of two phases, a first phase for forming a skeleton, and a second phase for filling up the skeleton, summarized as follows.

1. Constructing the skeleton of the assembly's structures by at most 7 perpendicular ribbons on C. By *frame*, we informally mean the intersection of a given plane with the surface of the cuboid, that wraps around it. Here, the planes P_X, P_Y and P_Z are being *framed* by several ribbons of tiles (forming the three sets R_X, R_Y and R_Z defined previously) during the assembly, and each step starts only when the previous step is finished.

   - R_X includes one ribbon for framing the first plane P_X.
   - R_Y includes two ribbons for framing the second plane P_Y.
   - R_Z includes zero or four ribbons (depending on the intersection of the two previous planes; details will be given later) for the frame of the third plane P_Z.

2. Filling the inside of the assembly's skeleton by distinctive tiles. In this step, the interior of the regions is partially filled by their distinctive tiles in a way that no connected component of the surface of C (as delimited by the three planes) has a neighbor with the same inner filling tile.

See Figure 7.10 for an illustration of the main elements of the assembly, when performed on a genus 0 cuboid.

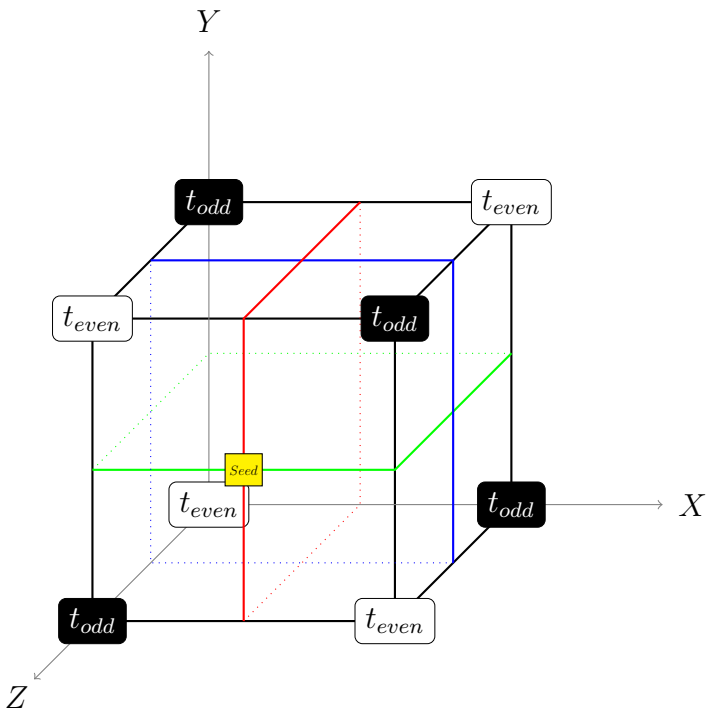


Figure 7.10: The skeleton of a $\mathcal{S}_\mathcal{G}$ assembly on an order-0 cuboid is shown in color. It is started from a seed (in yellow) and after the formation of the skeleton, the regions are partially filled by tiles of types t_{odd} and t_{even}.

Proof ideas

We now give an intuition of how Theorem 7.6 is proved. The key of the proof is the partition into regions using the three planes as described previously.

- The first case is when C has genus 0. Then the skeleton forms as intended, divides C into 8 disconnected regions (since there is no tunnel). Since the tiles of Y have labels occurring in tiles present in distant parts of the assembly only (and need to attach to two such tiles), no tile of Y can be part of the assembly in this case.
- The other case is when C has genus 1, that is, there is a tunnel. Assume that the skeleton has formed completely and has divided C into 8 initial regions, with two of them being identified due to the tunnel that connects two opposite sides of C. These two (initially distinct) regions will be filled with inner filling tiles of different types (t_{even} and t_{odd}). Moreover, the filling is done with stripes that are spaced differently. More precisely, the two spacings of the two regions (for the even and odd inner fillings) are coprime, in our case, 3 and 5. These coprime spacings ensure that, when the two inner fillings meet each other through the tunnel, not every stripe of the first region faces a stripe of the other region. (One stripe of a region might meet a stripe of the other region, but then for the next stripe, this does not happen.) Thus, in that case, there is always room for a tile of type t_{reg}, that will necessarily be present in the terminal assembly.

Special cases are when the skeleton passes inside the tunnel. If R_Z passes through the tunnel, then two ribbons of R_Z meet and a tile of type t_{mfs} will appear. If R_X or R_Y passes inside the tunnel, then R_Z is not formed and no inner filling exists. In this case, the ribbon of Y_{ew} tile types (part of R_Y) rebounds from east to west, parallel and between the support tiles; and ribbons of support tiles of the eastern and western middle finding systems meet each other. As a result, the tiles of types t_{ibc1} and t_{ibc2} appear between the tiles of type Y_{ew} and the western support tiles. (The details will be presented in upcoming sections.)

In order to simplify the explanation of the process of the assemblies, first, phase 1 is presented:

- how the skeleton grows depending on the placement of the seed;
- how the skeleton partitions C into distinct connected components;
- what its assigned region graph is.

Then, we study the phase of inner filling. Afterwards, we conclude with the proof of Theorem 7.6.

7.6 Description of terminal assemblies of $\mathcal{S}_\mathcal{G}$ on order-1 cuboids: $A_\square^{C_1}[\mathcal{S}_\mathcal{G}]$

For the study of the shape of the productions in O_1^t, the productions on O_0 will be useful as a reference. Hence first we show that $\mathcal{S}_\mathcal{G}$ partitions order-0 cuboids into eight distinct regions as presented in Section 7.3. Later we study the case of order-1 cuboids, that can have genus 1 or genus 0.

7.6.1 Terminal assemblies on order-0 cuboids: $A_\square^{C_0}[\mathcal{S}_\mathcal{G}]$

Now, we characterise the set $A_\square^{C_0}[\mathcal{S}_\mathcal{G}]$ of terminal assemblies on an order-0 cuboid C_0.

The structure of the skeleton

Lemma 7.10. *Let $C \in O_0$ be an order-0 cuboid and assume that the seed α_σ is placed at a normal placement $p \in Pl_N(C)$. Every terminal assembly of $\mathcal{S}_\mathcal{G}$ on C includes a "3-step skeleton" noted by $R_X \cup R_Y \cup R_Z$ where each part is located on the corresponding ribbon defined in Section 7.3.*

Proof. The assembly starts from the single seed tile $\sigma = (\epsilon, \epsilon, x', \epsilon)$ placed as a seed assembly α_σ. It has label x' with strength 2 at the south, and its other labels are ϵ with strength 0. In the first step, tiles make a vertical segment ribbon of tiles around C as the ribbon R_X, starting from the south of the seed and finishing at its north. More precisely, a tile of type $x' = (x', x''_e, x, x''_w)$ sits at σ's south and the label x gives rise to a vertical ribbon of tiles of type $x = (x, x_e, x, x_w)$ such that it grows around C and returns to the seed from the north. Moreover, two tiles of types $x'_e = (x'_e, x'_e, x_e, x''_e)$ and $x'_w = (x'_w, x''_w, x_w, x'_w)$ come at the east and the west of x'. By supporting the tile of type x, ribbons of tiles of types $x_e$and x_w form respectively in the south of x'_e and x'_w. Now, there is a ribbon made of three columns of tiles that build the ribbon R_X. See Figure 7.11 for an illustration, where the seed, shown in red, is in the center of R_X. The ribbon R_X divides C into

two regions, the *right side* and the *left side* of C with respect to the seed assembly α_σ. (We always assume that we view the cuboid from the point of view of the Z-axis, as in Figure 7.9, and thus *left*, *right*, *up*, *down*, *back*, *front* refer to this point of view.)

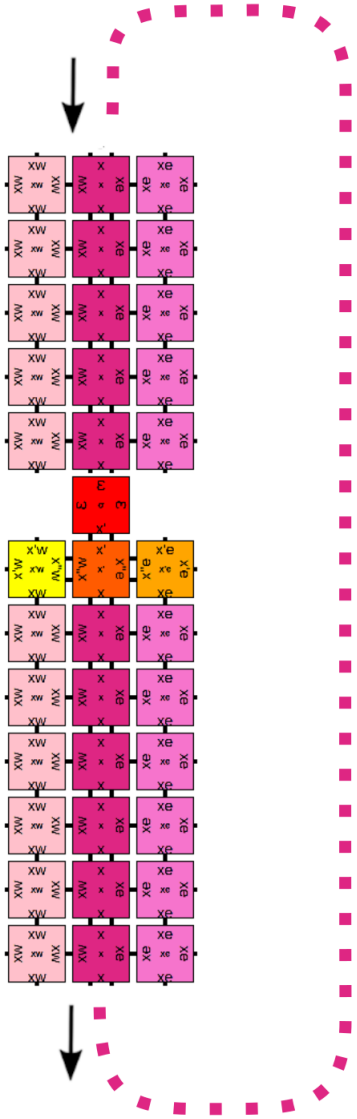


Figure 7.11: The ribbon R_X. The assembly starts from the south of the seed tile (in red at the center) and wraps around the order-1 cuboid.

Therefore, R_Y starts to form only when R_X rebounds to the north of the seed, using tiles of type y_e and y_w. The ribbons starting from both right and left sides of α_σ develop perpendicular to R_X by using four middle finding systems (defined in Lemma 6.8).

To elaborate, the tiles of type $y_e = (x_e, y_e, x'_e, \epsilon)$ and $y_w = (x_w, \epsilon, x'_w, y_w 1)$ sit in the assembly respectively between tiles of type x'_e and x_e at the east of R_X, and between tiles of type x'_w and x_w at the west of R_X. The tiles of type $y_{e1} = (\epsilon, y_{e1}, \epsilon, y_e)$ and $y_{w1} = (\epsilon, y_w, \epsilon, y_{w1})$ bind to y_e and y_w; and the tiles of type $y_{e2} = (\epsilon, , y_{e1}, \epsilon, y_{e2})$ and $y_{w2} = (\epsilon, y_{w1}, \epsilon, y_{w2})$ bind to the tiles of type y_{e1} and y_{w1}, respectively. Then, the tile types y_{ens} and y_{wns} appear next to y_{e2} and y_{w2}. From the east of y_{ens}, a ribbon of tiles of type Y_{ew} grows. Moreover, at the north and at south of y_{ens} and y_{wns}, the tiles of type y_{en}, y_{es}, y_{wn} and y_{ws} respectively appear, that are starting points for the eastern

and western IBC systems (Lemma 6.5) of the middle finding systems, that is, the seeds $\sigma^+_{en} = (over_{en}, ++_{en}, y{+}_{en}, \epsilon)$, $\sigma^+_{es} = (y{+}_{es}, ++_{es}, over_{es}, \epsilon)$; and $\sigma^+_{wn} = (over_{wn}, \epsilon, y{+}_{en}, ++_{wn})$, $\sigma^+_{ws} = (y{+}_{ws}, \epsilon, over_{ws}, ++_{ws})$ start four IBC systems $IBC1_{en}$, $IBC1_{es}$ (east), and $IBC1_{wn}$, $IBC1_{ws}$ (west) with supports $++_{en} = (1_{en}, ++_{en}, en, ++_{en})$ and $++_{es} = (es, ++_{es}, 1_{es}, ++_{es})$, and $++_{wn} = (1_{wn}, ++_{wn}, wn, ++_{ewn})$ and $++_{ws} = (ws, ++_{ws}, 1_{ws}, ++_s)$ respectively. Note that each system has its own distinguished tile types. See Figure 7.12 for an illustration of the growth of R_Y after the completion of R_X.

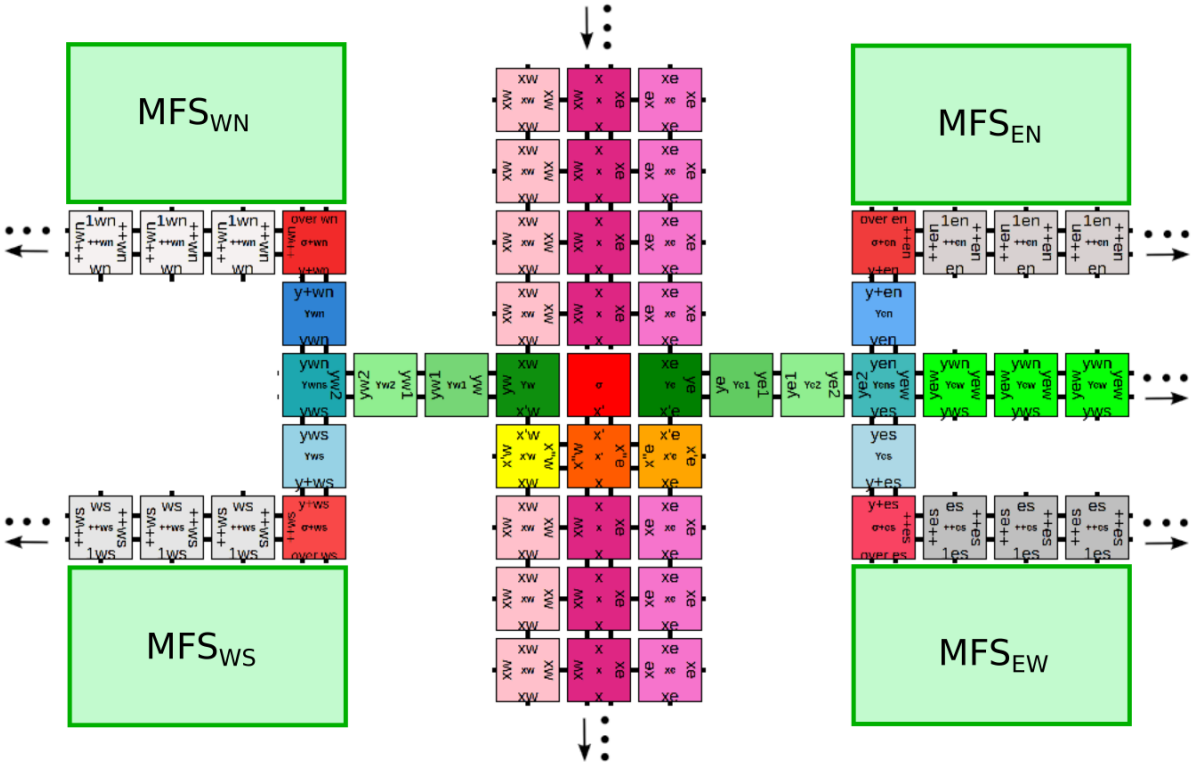


Figure 7.12: The formation of the R_Y ribbon (horizontal), out of R_X (vertical). The initial seed of the assembly is in the center (in red). When R_X is finished, from the east and the west of the seed the starter tiles of R_Y appear. Then, four middle finding systems MFS_{EN}, MFS_{ES}, MFS_{WS}, and MFS_{WE}, each with distinct labels, start to form respectively from four red seeds $\sigma + +_{EN}$, $\sigma + +_{ES}$, $\sigma + +_{WN}$, $\sigma + +_{WS}$.

After that, the ribbons of the four $IBC1$ systems in R_Y meet the R_X ribbon for the second time. See Figure 7.13. Then, the tiles of types x_e and x_w of R_X act as finishing block tiles for the middle finding system.

Then, the four tiles of types $t_{eu} = (1_s, \epsilon, x_e, 1)$, $t_{ed} = (1_s, 1, x_e, \epsilon)$, $t_{wu} = (1_s, 1, x_w, \epsilon)$ and $t_{wd} = (1_s, \epsilon, x_w, 1)$ respectively, appear as row tile number of 1 for the second set of IBC systems $IBC2_{en}$, $IBC2_{es}$ (east), and $IBC2_{wn}$, $IBC2_{ws}$ (west) of the middle finding systems. Note that the tiles of the $IBC2$ systems also have different labels from the ones of the $IBC1$ systems (except the exterior labels of the most significant bit tiles in the $IBC1$ systems). On the same underlying rectangle as the underlying rectangle of R_Y, the ribbons grow and they stop in the middle of R_Y's ribbons.

Once the R_Y ribbons form, they separate C into an *up side* and a *down side*. Thus, C is now partitioned into four separate regions due to the first and second step ribbons.

Next, by finding the middle of each of R_Y's ribbons on the right and left faces, four new perpendicular ribbons are generated from the tiles of type $t_{ml} = (1, z''_l, 0', 1)$ at the left-up

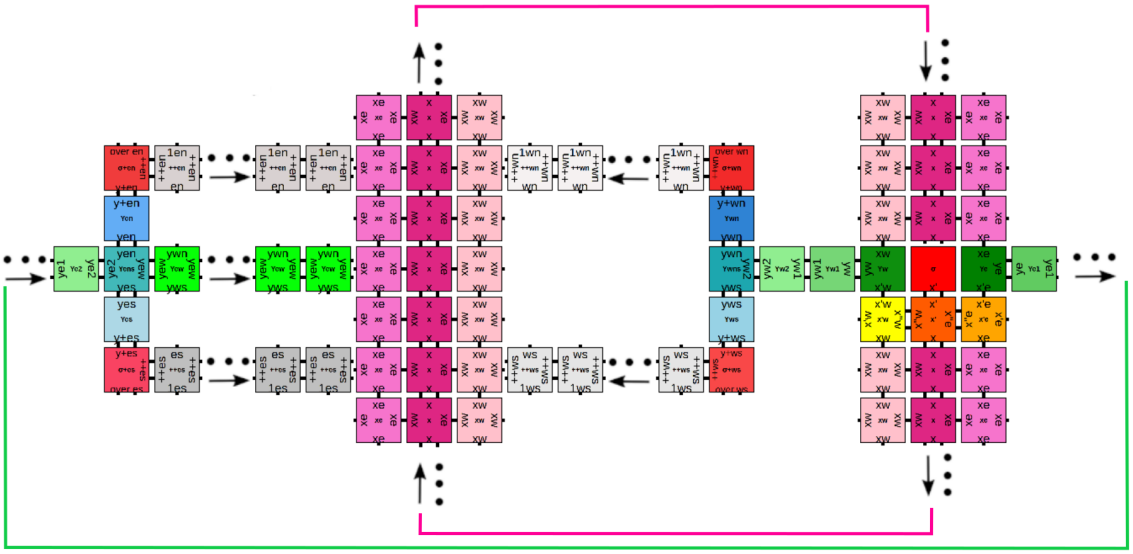


Figure 7.13: The second collision of R_Y and R_X.

and right-down sides, and $t_{mr} = (1, 1, 0', z_r'')$ at the right-up an left-down sides. They are the replacements of the tile of type t_m of the middle finding systems. See Figure 7.14 for an illustration. They go on until they reach R_X on the upper and down faces (see Figure 7.15). The union of these four ribbons forms a frame for the plane P_Z. This step creates a separation between the *front side* and the *back side* of the cuboid C with respect to α_σ. Note that all the ribbons are able to form completely because by our hypothesis, the seed is located on a normal placement. This is an essential condition since we need enough space for using counters in the assembly. We show the detailed assembly of R_X, R_Y and R_Z in Figure 7.16 that starts from the seed tile α_σ (in red) at the left of the Figure where R_X rebounds on C. The two western middle finding systems and the parts of R_Z that grow from them are omitted in the figure, however they are simply the mirror images of the eastern ones that are shown in the figure.

Similar to Section 7.3, these three steps partition C into 8 distinct regions. Note that, because the seed started in a normal placement, there is enough space for this assembly. Indeed, a normal placement implies that there are at least $6\lceil\log(N)\rceil + 9$ free tile spaces around the seed (on the same face of the cuboid where the seed is placed).

Our assembly needs at most $6\lceil\log(N)\rceil + 9$ tile spaces. Firstly, it needs $\lceil\log(N)\rceil$ space for each of $IBC1$ systems and $IBC2$ systems, and DBC systems; for each of the two middle finding systems. Moreover, 9 further tile spaces are needed: one for Y_{ew} tiles, 2 for the support of middle finding systems, two for the space between them, and 2 other for each ribbon R_Z. □

Inner filling of the skeleton After the formation of the skeleton, the second phase is to fill the eight regions by lines of interior tiles, once the R_Z ribbons reach R_X. Since the region graph of an order-0 cuboid C is a 2-colorable (i.e. bipartite) graph, we use two tile types to distinctly tile C.

Lemma 7.11. *Let $C \in O_0$ be an order-0 cuboid. For all the terminal assemblies $\alpha \in A_\square^{C_0}[\mathcal{S}_\mathcal{G}]$ started from a seed $\sigma \in Pl_N(C)$, the tile $t_{even} = (z_e, x_{ev}, z_e, x_{ev})$ appears in the even regions and the tile $t_{odd} = (z_o, x_{od}, z_o, x_{od})$ appears in the odd regions.*

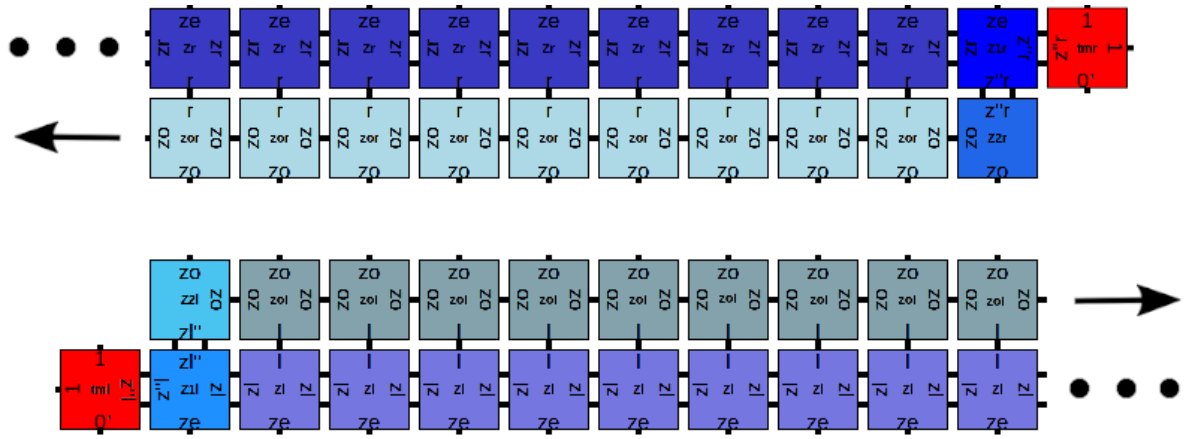


Figure 7.14: Two ribbons of R_Z.

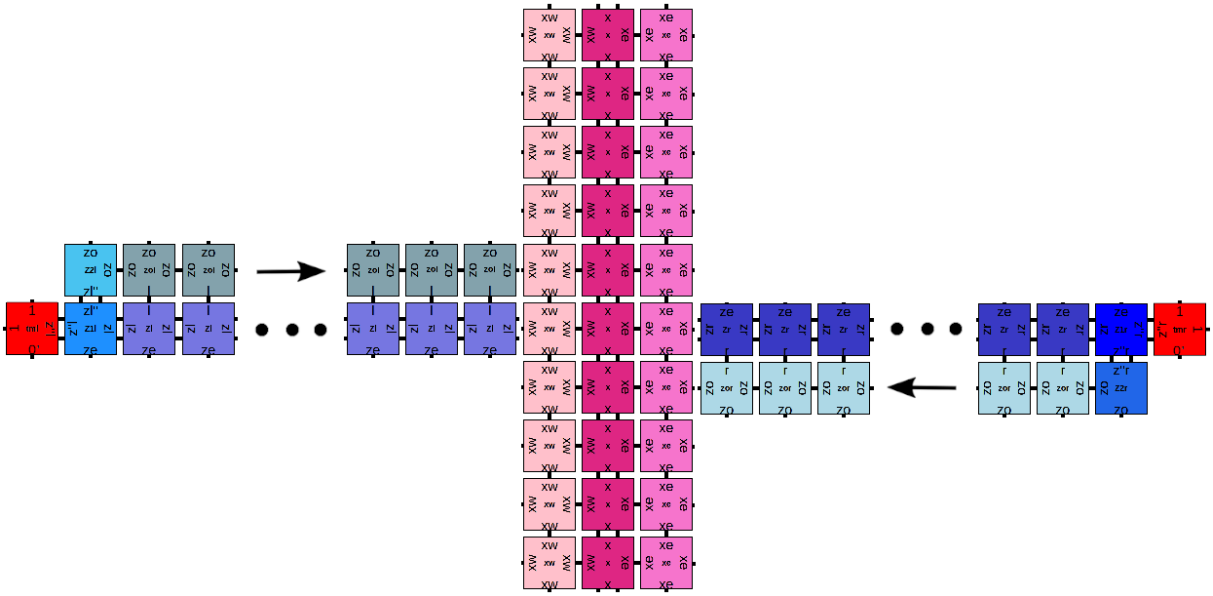


Figure 7.15: Two ribbons of R_Z meeting the ribbons of R_X.

Proof. For an assembly that is started from a seed in a normal placement, by Lemma 7.10, the cuboid C is partitioned by a skeleton into odd and even regions. First, four ribbons of tiles appear at the intersection of the R_X and R_Z ribbons. These ribbons are formed by tiles of the types of Figure 7.8. From the parts along R_X, straight lines of tiles start growing parallel to the x axis using strength 2 glues x_{ev} and x_{od}. Thanks to modulo 5 (resp. 3) counters on the even (resp. odd) R_X border tiles, there is one such line every other 5 (resp. 3) position along that part of the border with tiles of type $t_{even} = (z_e, x_{ev}, z_e, x_{ev})$ (resp. $t_{odd} = (z_o, x_{od}, z_o, x_{od})$). These lines form the even (resp. odd) filling tiles and fill the partitioned regions. See Figure 7.17 for an illustration of the assembly. □

The 2-coloring of G_C indicates also where the tiles of type t_{even} and of type t_{odd} can be placed. Therefore, the regions R_{XYZ} $(X, Y, Z \in \{0, 1\})$ are tiled with t_{even} if and only if the sum $X + Y + Z$ is even, and the regions with odd sum are covered by t_{odd}. For more clarity, see Figure 7.3 where the regions corresponding to t_{even} are colored white and the ones with t_{odd} are colored black in the region graph G_C of the order-0 cuboid C.

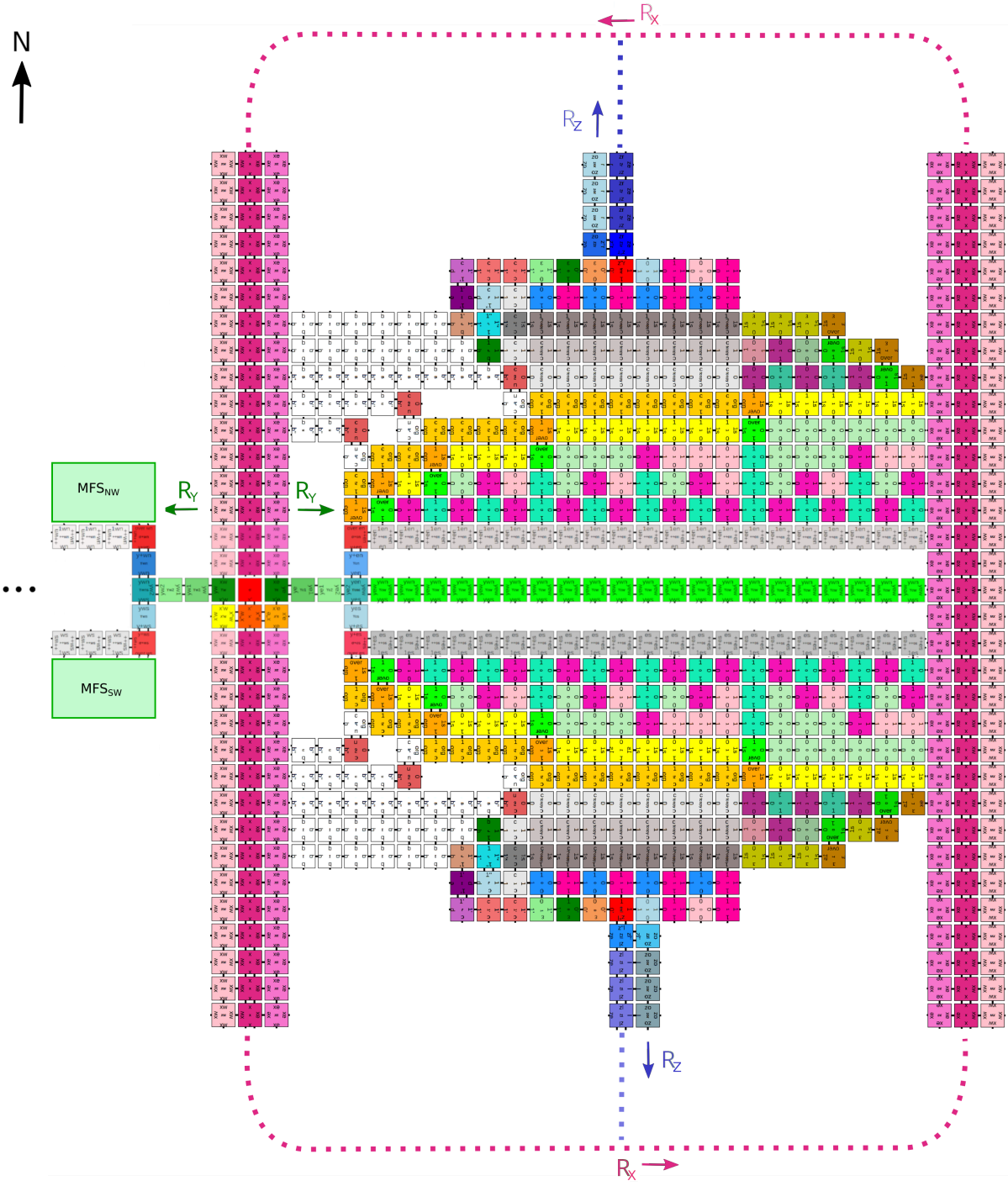
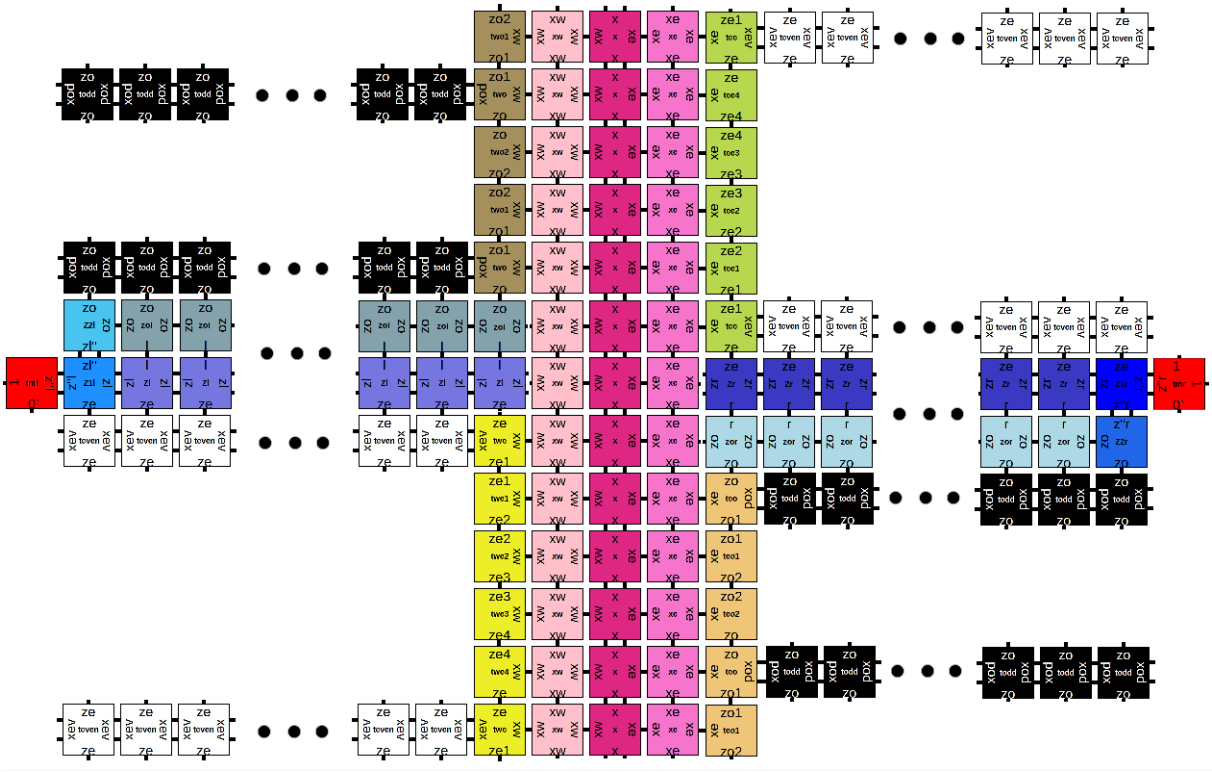
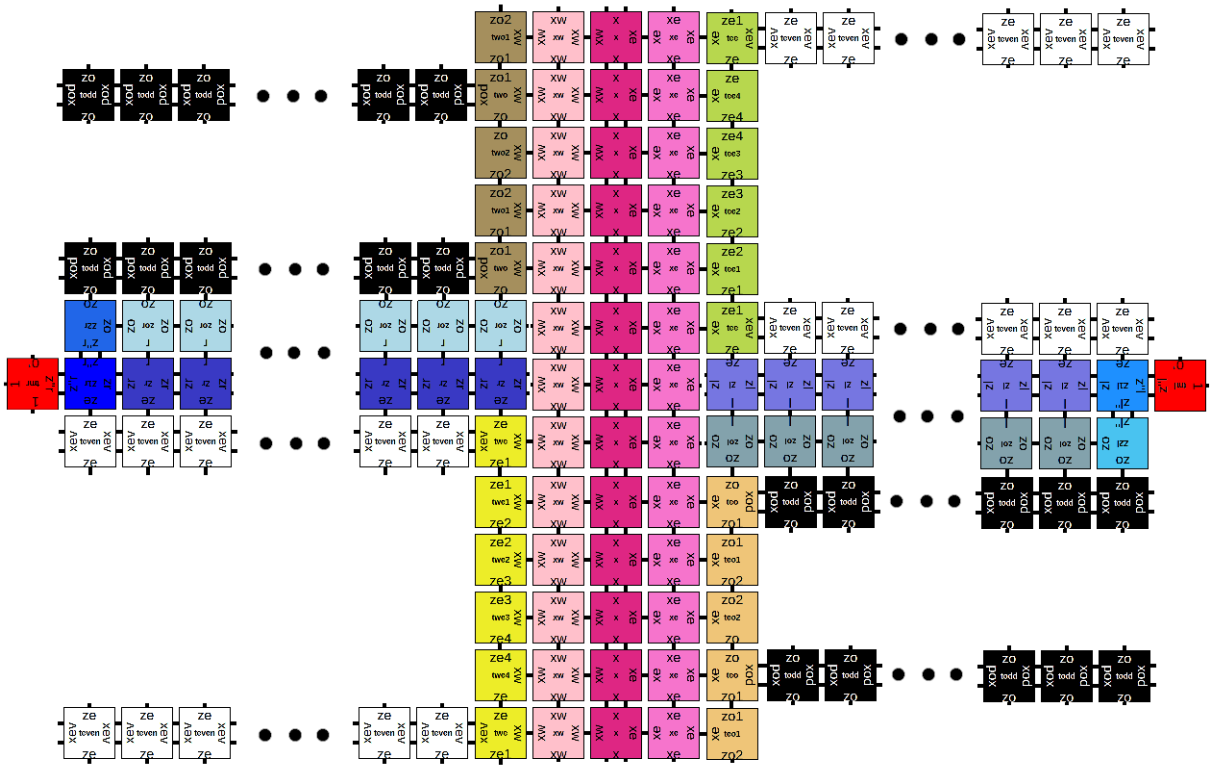


Figure 7.16: The assembly of R_X, R_Y and R_Z on an order-0 cuboid. The seed is located in the middle of R_X, on the left. R_X grows from the south of the seed and finishes at its north. Then, R_Y grows, including northern and southern middle finding systems, and a ribbon of green y_{ew} tiles between them that ends by arriving at P_X. At the end, R_Z starts to assemble from the found middle tile of R_Y (in red) and finishes by arriving at R_X. Note that The western middle finding systems and its assigned parts of R_Z are omitted for the sake of brevity, however they are the mirror image of the eastern ones, excluding y_{ew} ribbon of tiles.



(a) Up side collision of R_X and R_Z.



(b) Down side collision of R_X and R_Z.

Figure 7.17: The inner filling with tiles of types t_{even} (white) and t_{odd} (black) at the two places where the R_Z ribbons meet R_X.

7.6.2 Terminal assemblies on order-1 cuboids with genus 1 : $A_\square^{C_t}[\mathcal{S}_\mathcal{G}]$

We consider now in detail the process of the assembly of $\mathcal{S}_\mathcal{G}$ for order-1 cuboids with a tunnel. This section will characterise the set $A_\square^{C_t}[\mathcal{S}_\mathcal{G}]$ of terminal assemblies on such order-1 cuboid C_t. Let C be an order-1 cuboid. The key element of the proof is the appearance of some specific tile in each assembly when it has less than 8 regions. The assemblies on C have a skeleton with a different shape depending on the region graph associated with the placement of the seed. Let P_i and R_i for $i \in \{X, Y, Z\}$ be defined as presented in Section 7.3. If a plane P_i intersects along the width of the tunnel, it acts like a separator between the two parallel faces where the tunnel's entrances are located. If a plane P_i intersects along the length of the tunnel, the tiles of R_i enter and pass inside the tunnel. Moreover, three types of partitions into regions are possible and the possible numbers of regions are: 7 regions when one plane intersects along the width of the tunnel, 5 regions when one plane intersects along the length of the tunnel and one along the width, and 1 region when three perpendicular planes intersect along the tunnel, one along the width and the others along the length.

Depending on each case, the presence of some tiles from $Y = \{t_{reg}, t_{mfs}\} \cup T_{ibc} \subseteq T$ in assemblies of $\mathcal{S}_\mathcal{G}$ witnesses the presence of a tunnel. In Figure 7.26, the places where the tunnel implies the presence of each of these tiles are demonstrated.

Note that in the following, $G_C(\sigma)$ refers to the region graph $G_C(p)$ such that p is the position of the seed α_σ on C.

Case 1 (7 regions): one plane intersects along the width of the tunnel.

In this case, tiles of types t_{odd} and t_{even} touch, which enforces the attachment of t_{reg} or t_{mfs}.

At least one of the planes P_X, P_Y and P_Z introduced in Section 7.3 intersects with the tunnel of C, since its entrances are on parallel faces of the cuboid, and these planes are located between parallel faces. When the tunnel has an intersection with only one of the three planes, the plane intersects along the width of the tunnel. For example, in Figure 7.18, the tunnel has an intersection with the plane P_X only.

Lemma 7.12. *Let $C = C_0 \setminus C'_0 \in O_1^t$ be an order-1 cuboid with the dimensions at least 10 for C'_0. Assume that the seed α_σ is placed in a normal placement $p \in Pl(C)$. In a terminal assembly of the system $S_\mathcal{G}$, if only one of the planes defined in Section 7.3 intersect with the tunnel, $G_C(\sigma)$ has 7 regions and a tile of type t_{reg} or t_{mfs} appears in the assembly.*

Proof. Let the seed be placed in a manner that only one of the planes P_X, P_Y or P_Z intersects along the width of the tunnel. The plane that intersects the tunnel is the separating buffer of two regions R_{xyz} and $R_{x'y'z'}$ containing the two tunnel's entrances. In this case, the two regions R_{xyz} and $R_{x'y'z'}$ get combined into a single region via the tunnel. Therefore, the number of distinct regions decreases to 7 regions. See Figure 7.18 for an illustration.

Without loss of generality, assume that $x + y + z$ is an odd number and $x' + y' + z'$ is an even number. When two regions R_{xyz} and $R_{x'y'z'}$ are joined by the tunnel, tiles of type $t_{odd} = (z_o, x_{od}, z_o, x_{od})$ from R_{xyz} and of type $t_{even} = (z_e, x_{ev}, z_e, x_{ev})$ from $R_{x'y'z'}$ both exist in the new unique region. See Figure 7.17. We show that the tile type $t_{reg} = (z_e, \epsilon, z_o, \epsilon)$ or t_{mfs} must then occur in the assembly. Indeed, the tile types t_{reg} and t_{mfs} are the only tile types of $\mathcal{S}_\mathcal{G}$ with labels z_o and z_e of inner filling tiles t_{odd} and t_{even}. In Figure 7.23, the

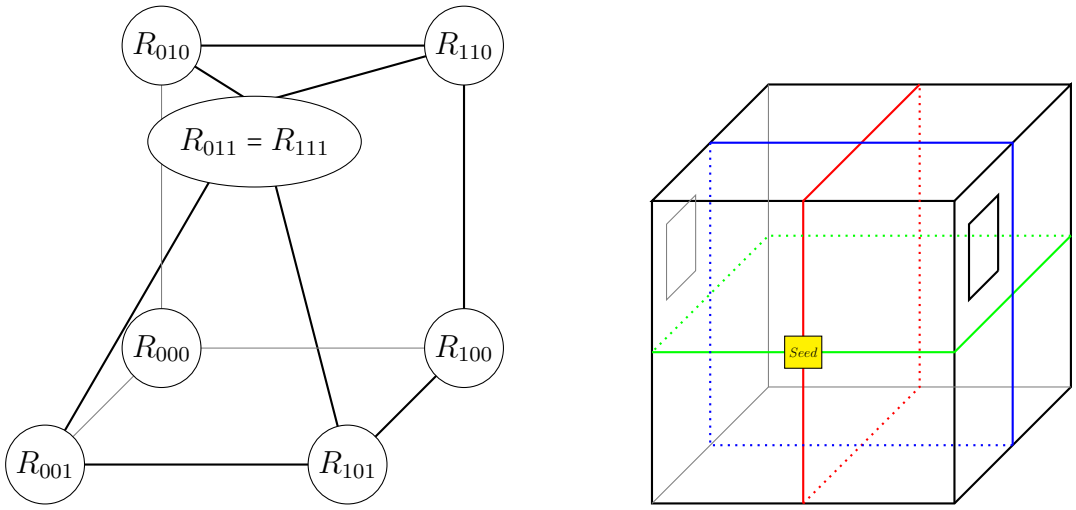
places on C that reveal a tunnel by t_{reg} or t_{mfs} are shown. To conclude the proof, one needs to show that in a region with a disconnected border, there is a *good* empty space, that is an empty space which sees both an even tile and an odd tile through strength 1 sides. Then, this space can be filled by neither type of filling tiles, but it must eventually be filled by a tile of type $t_{reg} = (z_e, \epsilon, z_o, \epsilon)$ or $t_{mfs} = (z_e, z_o, z_e, z_o)$. In a region with a tunnel, on each side of the tunnel, the border of every 10×10 square must be crossed by either

- at least two of the lines of tiles starting from R_X on that side of the tunnel, or
- at least two of the lines exiting the tunnel.

In particular, because C_0' is at least 10×10 units wide, there are at least two lines crossing one of the edges the tunnel in the same direction. Each such line must either reach the opposite connected component of the border, be stopped orthogonally by a line from the opposite side of the tunnel, or run head-first into an opposite line. Consider such a pair of lines, with minimal distance between them. In particular, that distance must be at most 10.

- If one of the lines reaches the opposite connected component of the border, either of the spaces next to its end is *good* and in this case the tile of type t_{reg} appears in the assembly;
- likewise, if one of them is stopped orthogonally by a line from the opposite side of the tunnel, one of the spaces next to the intersections is *good* and a tile of type t_{mfs} appears.

Moreover, if one of them runs head-first into an opposite line, the other cannot, because their distance cannot be at the same time divisible by 15, positive and less than 10. Hence the pair satisfies one the previous cases. This concludes the proof of that case of our construction. □



(a) The region graph

(b) The cuboid and the three planes.

Figure 7.18: The case where $C \in O_1^t$ is partitioned into 7 distinct regions. If there is a tunnel between two distinct regions, a tile of type t_{reg} or t_{mfs}, which have common labels with both t_{even} and t_{odd}, must appear in the assembly.

Case 2 (5 regions): the tunnel intersects with P_Z, and exactly one of P_X and P_Y.

Lemma 7.13. *Let $C \in O_1^t$ be an order-1 cuboid and assume that the seed α_σ is placed in a normal placement $p \in Pl(C)$. In a terminal assembly of the system $S_\mathcal{G}$, if the plane P_Z and exactly one of the planes P_X and P_Y defined in Section 7.3 have an intersection with the tunnel, there exist 5 regions on the cuboid and a tile of type t_{mfs} appears in the assembly.*

Proof. If the seed is placed where the tunnel has intersection with two perpendicular planes, one of them intersects the tunnel along its width and the other one along its length. If P_Z intersects with the tunnel along the length, the ribbons of R_Z meet each other inside the tunnel. However, if P_Z intersects the tunnel along its width, they meet outside the tunnel.

In both cases, the tile $t_{mfs} = (z_e, z_o, z_e, z_o)$ appears in the assembly when two frame ribbons of P_Z meet each other. Note that when the tunnel has intersection with P_Z and one of the planes P_X or P_Y, the cuboid is separated into two connected components such that one of them is a cuboid with genus 0 and the other one is a cuboid with genus 1. The part with genus 0 has 4 distinct regions, and the part with genus 1 (containing a tile of type t_{mfs}) has one single region. In total, there exist 5 distinct regions on the cuboid C. For an illustration of the skeleton and its graph in this case, see Figure 7.21. □

Case 3 (1 region): the tunnel intersects with P_X and P_Y.

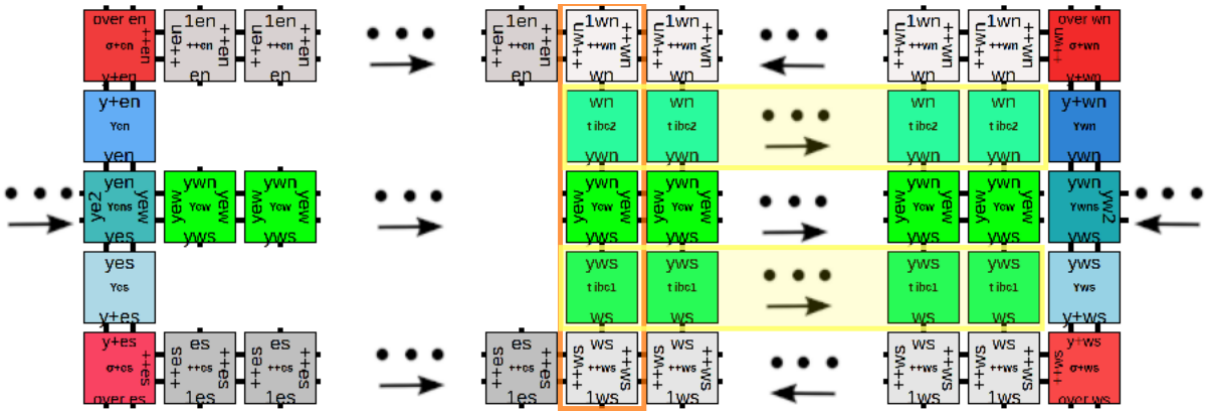
Lemma 7.14. *Let $C \in O_1^t$ be an order-1 cuboid and assume that the seed α_σ is placed in a normal placement $p \in Pl(C)$. In a terminal assembly of the system $S_\mathcal{G}$, if the two planes P_X and P_Y defined in Section 7.3 intersect the tunnel, there exists one region on the cuboid and a tile of type $T_{ibc} = \{t_{ibc1}, t_{ibc2}\}$ appears in the assembly.*

Proof. In this case, the skeleton of the assembly is not the same as before. Recall the process of the assembly's skeleton: the ribbon R_X is generated independently from α_σ. Two segment ribbons of R_Y begin to grow after rebounding on R_X, regardless of passing through a tunnel or not. However, the ribbons of R_Z start to grow only after finding the middle of R_Y and they end by reaching the ribbon of R_X. Considering this process, when the two planes P_X and P_Y intersect with the tunnel, the plane P_Z is not able to form since there is a tunnel that does not permit to have the second collision of R_Y and R_X. Therefore, the process of finding the middle of R_Y is not able to continue and the ribbons of R_Z are not able to form.

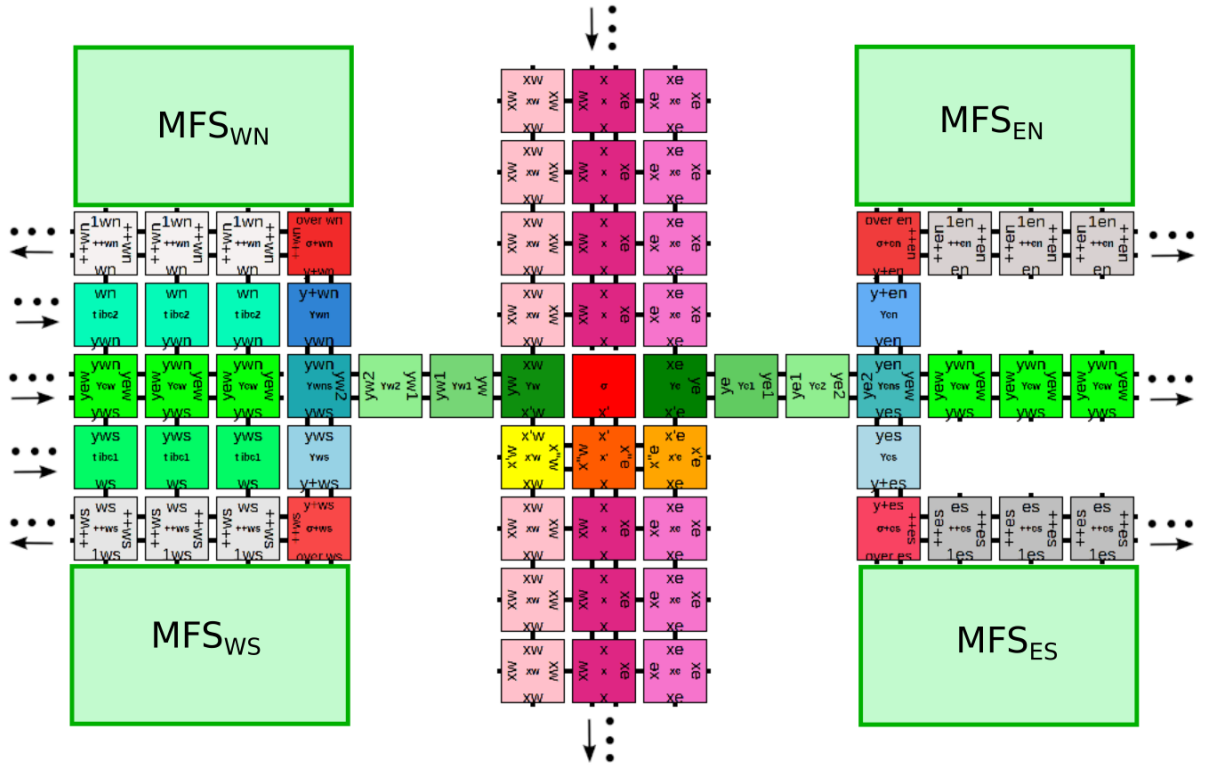
Moreover, the eastern and western parts of R_Y meet each other and tiles of type of T_{ibc} appear there. This happens inside the tunnel if P_Y intersects the tunnel along its length, and outside the tunnel if it intersects the tunnel along its width. In either case, a tile of one of the T_{ibc} types appears. See Figure 7.19 for an illustration of the assembly in this case, and Figure 7.25 for an illustration of the places where the presence of a tunnel entrance implies that we have a tile of one of the T_{ibc} types.

Note that the skeleton consists of two closed loops of R_X ribbons and R_Y ribbons. This phenomenon demonstrates that the genus of C is 1. In order to have a better overview, see Figure 7.20. Furthermore, there is only one single region throughout the whole surface of C. □

Note that the situation when the seed is located inside the tunnel is similar to Case 3, up to topological isomorphism.



(a) When the genus of the order-1 cuboid is 1, there is no second collision between R_X and R_Y. In this case, the support tiles of the eastern and western middle finding systems meet each other. As a result, the tiles of types t_{ibc1} and t_{ibc2} appear between the tile types of Y_{ew} and the western support tiles. The orange rectangle shows the first time that tiles of types t_{ibc1} and t_{ibc2} appear in the assembly, and the yellow rectangles show the tiles of types t_{ibc1} and t_{ibc2} in the assembly.

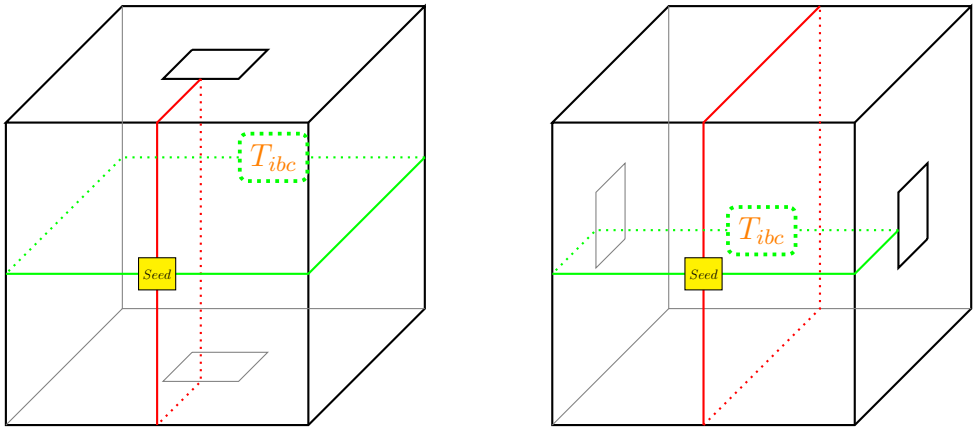


(b) R_X and R_Y when the genus of the order-1 cuboid is 1. In this case, the ribbon of tiles of type Y_{ew} meets the western tiles, and so the tile types t_{ibc1} and t_{ibc2} appear in the assembly, witnessing that the genus of the order-1 cuboid is 1.

Figure 7.19: The R_Y ribbons when the genus of the order-1 cuboid C is 1.

From Lemmas 7.12, 7.13 and 7.14, the following corollary is obtained:

Corollary 7.15. *Let $C = C_0 \setminus C_0' \in O_1$ be an order-1 cuboid with the dimensions at least 10 for C_0' and α be an assembly of the TAS $\mathcal{S}_\mathcal{G} = (\Sigma, T, \sigma, str, \tau)$ such that its seed is placed at a normal placement. If there is a tunnel on C (i.e. its genus is 1), at least one tile type from $Y = \{t_{reg}, t_{mfs}, t_{ibc1}, t_{ibc1}\} \subseteq T$ exists in all terminal assemblies of $\mathcal{S}_\mathcal{G}$ on C.*



(a) The tunnel intersects along the length of plane P_X and width of plane P_Y.

(b) The tunnel intersects along the width of plane P_X and length of plane P_Y.

Figure 7.20: Intersection of tunnel with two planes P_X (red) and P_Y (green).

Proof. If there is a tunnel on C, at least one of the planes P_X, P_Y and P_Z defined in Section 7.3 intersects with the tunnel since its entrances are on parallel faces of the cuboid, and these planes are located between parallel faces.

First, if the tunnel of C intersects with only one of the planes, due to Lemma 7.12, a tile of type t_{reg} or t_{mfs}, which are the only tile types of $\mathcal{S}_\mathcal{G}$ with labels in common with both inner filling tile types t_{odd} and t_{even}, appear in the assembly. In Figure 7.23, the places where the presence of these tile types displays the presence of the tunnel is shown.

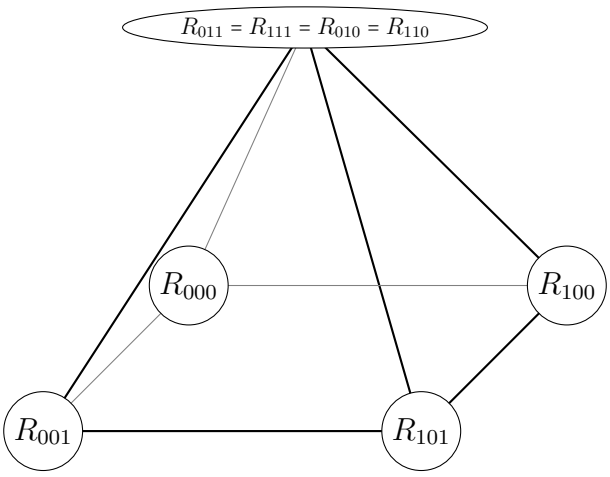
Next, if two planes (among them P_Z) intersect with the tunnel on C, a tile of type t_{mfs} appears in all terminal assemblies on C by Lemma 7.13. See Figure 7.24 for the places where the presence of a tile of type t_{mfs} displays the presence of the tunnel.

Finally, if the two planes P_X and P_Y intersect with the tunnel, Lemma 7.13 implies that tile types of the set $T_{ibc} = \{t_{ibc1}, t_{ibc2}\}$ are present in the assembly. See Figure 7.25 for the places where the presence of a tunnel implies the presence of a tile of one of the T_{ibc} types. □

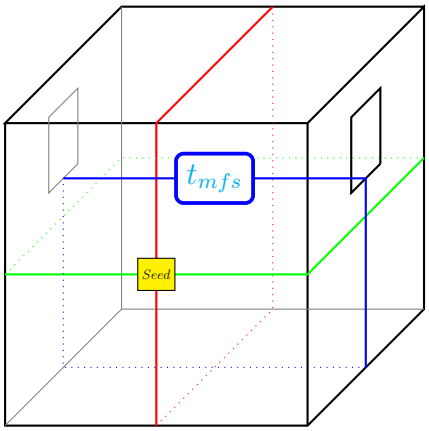
The places where a tunnel implies the presence of a tile of Y are shown in Figure 7.26.

7.6.3 Terminal assemblies on order-1 cuboids with genus 0 : $A_\square^{C_c}[\mathcal{S}_\mathcal{G}]$ and $A_\square^{C_p}[\mathcal{S}_\mathcal{G}]$

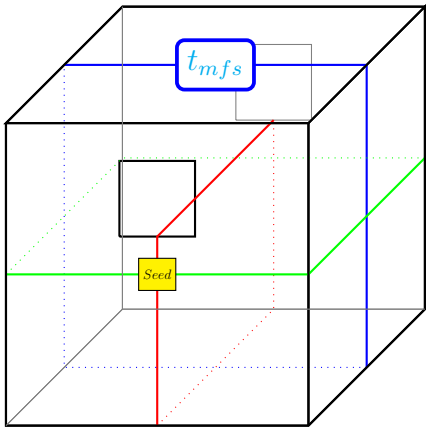
Now notice on the cases that $\mathcal{S}_\mathcal{G}$ assembles are on an order-1 cuboid $C \in O_1^c$ (the order-1 cuboids with concavity whose genus is 0), or $C \in O_1^p$ (the order-1 cuboids with a pit whose genus is 0). In these cases, the assembly's process is similar to the assembly on order-0 cuboids. The frame ribbons form completely by the assumption that the seed is located on a normal placement of C, the assembly's skeleton is formed completely and separates C into 8 distinct regions, and the insides of the regions are tiled independently by inner filling lines of tiles of types t_{odd} and t_{even}. However, in the case of O_1^c, the regions do not necessarily meet edge to edge, see Figure 7.22 for an illustration.



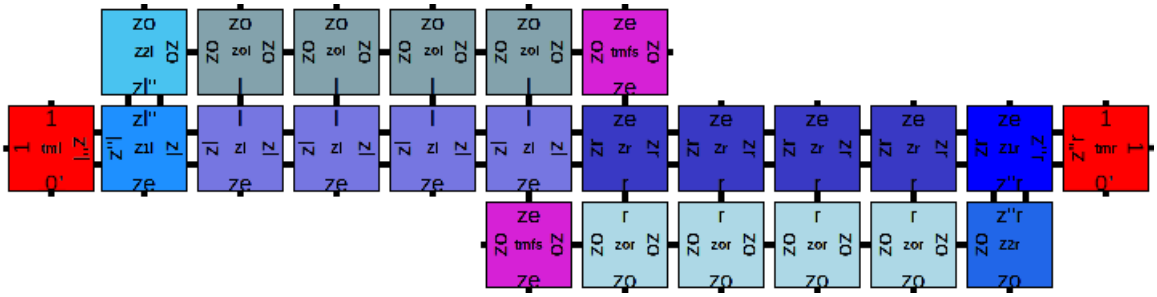
(a) The region graph



(b) The tunnel intersects along the width of plane P_X and length of plane P_Z.



(c) The tunnel intersects along the length of plane P_X and width of plane P_Z.



(d) A tile of type t_{mfs} appears if and only if two segments of R_Z (in purple) intersect each other by passing through a tunnel (instead of reaching R_X).

Figure 7.21: The case where $C \in O_1^t$ and C is partitioned into 5 distinct regions.

7.7 Detecting the genus of order-1 cuboids via $\mathcal{S}_{\mathcal{G}}$: proof of the main theorem

Before proving Theorem 7.6, we need to prove following lemma:

Lemma 7.16. *Let C be an order-1 cuboid. If one tile of $Y=\{t_{reg}\}\cup\{t_{mfs}\}\cup T_{ibc} \subseteq T$ exists*

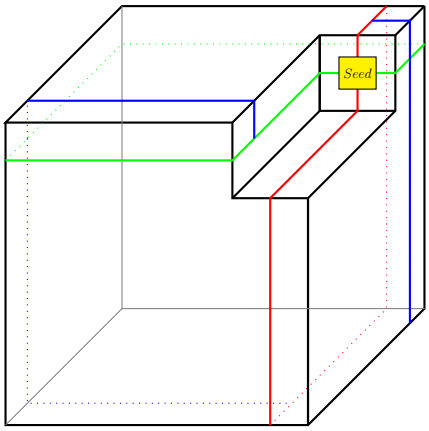


Figure 7.22: Cuboid with concavity: The three planes P_X (red), P_Y (green) and P_Z (blue), which consists of two semi-planes.)

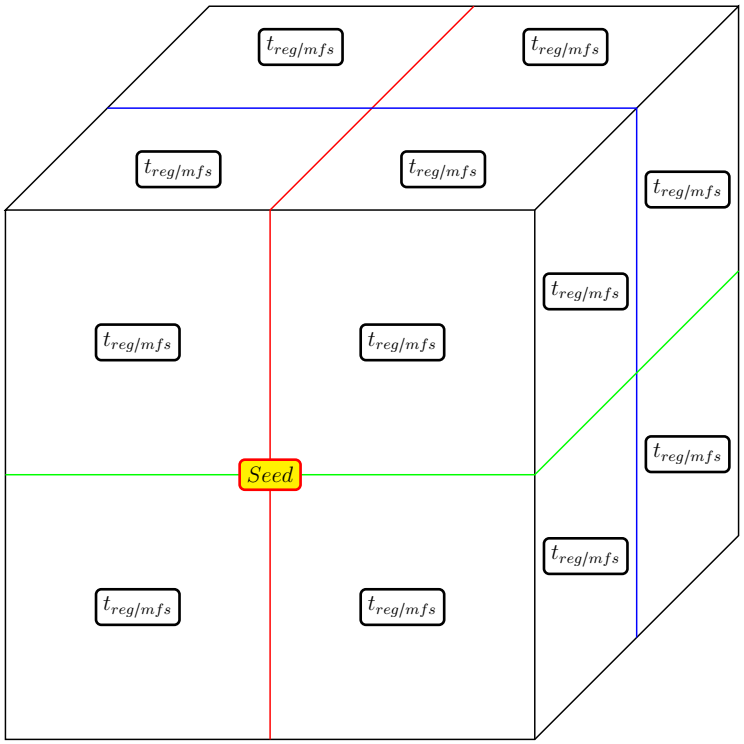


Figure 7.23: The places on an order-1 cuboid where, if a tunnel is placed there, a tile of type t_{reg} ou t_{mfs} appears.

in a terminal assembly of $\mathcal{S}_\mathcal{G}$ on C starting from a seed in a normal placement, there is a tunnel on C.

Proof. Firstly, if a tile of type t_{reg} is in the terminal assembly, its common labels with both inner filling tile types t_{odd} and t_{even} shows that at least two regions are connected i.e. there are at most 7 distinct regions on C. Recall that by Section 7.6.1, a terminal assembly of $\mathcal{S}_\mathcal{G}$ on an order-1 cuboid with genus 0 partitions the cuboid into 8 distinct regions. Therefore, C cannot have genus 0 and there is a tunnel between these two regions.

Secondly, if a tile of type t_{mfs} exists in a terminal assembly on C, two cases are possible. In one case there is a tunnel that intersect only P_X with width and t_{even} and

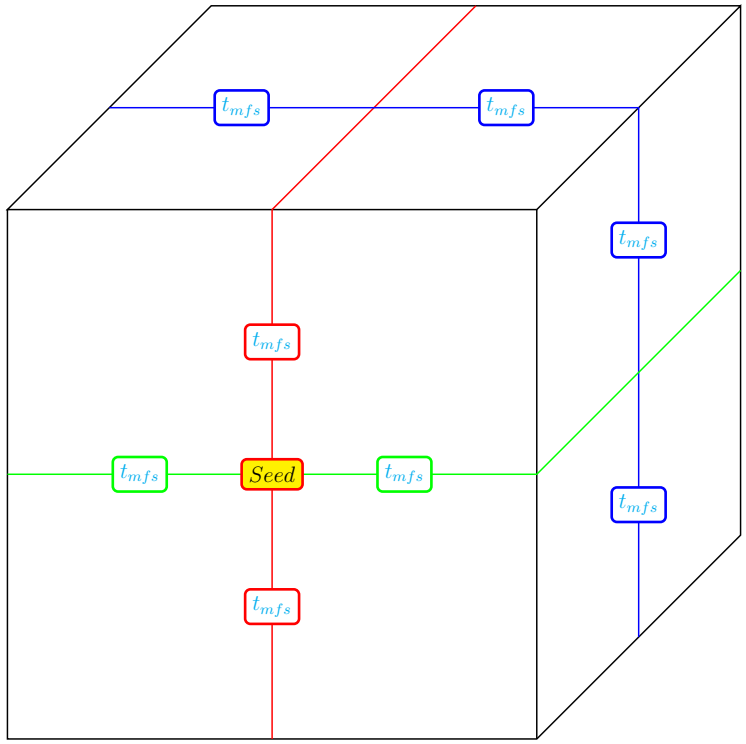


Figure 7.24: The places on an order-1 cuboid where, if a tunnel is placed there, a tile of type t_{mfs} appears.

t_{odd} intersect perpendicularly each other and as a result t_{mfs} appears in the assembly. In the other case, two ribbons of R_Z must meet each other since the tiles whose labels correspond to the labels of t_{mfs} are those of the R_Z ribbons. Recall that the R_X and R_Y ribbons intersect at two places: one at the seed (since R_Y grows out of R_X) and a second time, where the tiles of type t_{eu}, t_{ed}, t_{wu} or t_{wd} appear in the assembly as the row tile number 1, in the second IBC system of the middle finding systems. The two ribbons of R_Z, together with the parts of the middle finding system located between the second intersection of R_X and R_Y on the one hand, and R_Z on the other hand, form a closed ribbon on the surface of C (highlighted in green and blue Figure 7.27). This ribbon and R_X pass through each other perpendicularly at only one place. Since they pass through each other perpendicularly, it can be concluded that the cuboid C cannot be topologically homeomorphic to the sphere, or in other words, be a genus 0 cuboid and a tunnel must exist.

Lastly, assume that a tile of $T_{ibc} = \{t_{ibc1}, t_{ibc2}\}$ types appears in the assembly. Note that these tiles has only two labels with strength 1. One of two labels of each tile types of T_{ibc} tile types is in common with label of Y_{ew} that grow from the eastern parts of R_Y, and the other one is as those of the labels of tiles of western $IBC1$ systems in R_Y. Since the eastern and western ribbons are located opposite of each other Therefore, the ribbons of R_Y must collide. As a result there is a tile of one of the T_{ibc} types in the assembly and they do not reach the R_X. Thus the ribbons R_X and R_Y have no intersection except at the seed. Since they pass through each other perpendicularly, as in the previous case, C cannot be topologically homeomorphic to a sphere. Therefore, a tunnel must exist so that R_X and R_Y do not intersect in two places. □

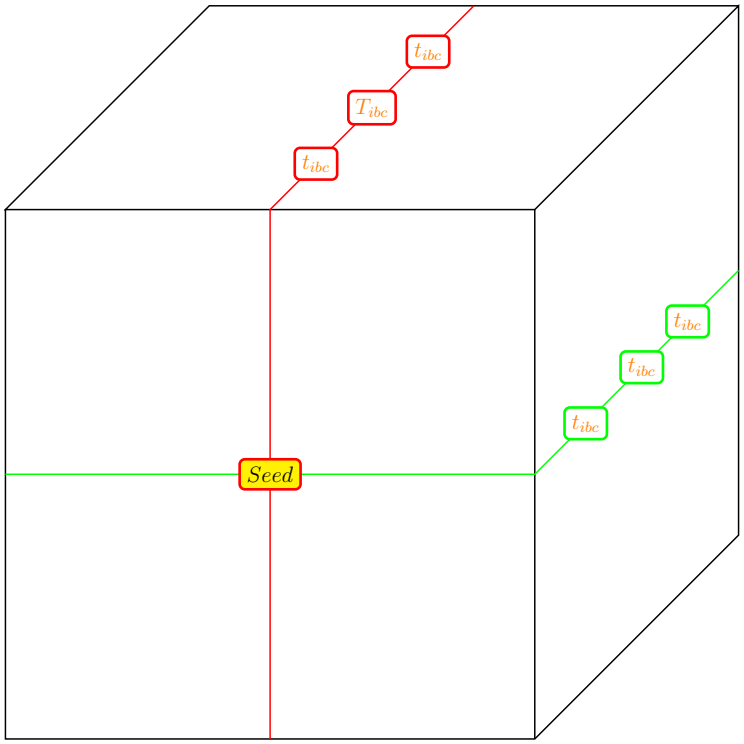


Figure 7.25: The places on C where t_{ibc} displays the presence of a tunnel on C. Note that the tunnel appears by t_{ibc} also when the seed is inside the tunnel, since up to topological isomorphism, it is the same case

Proof of the Main Theorem

We are now ready to prove Theorem 7.6. Informally, the general principle of the construction is as follows: cut the order-1 cuboid into regions and check if the partition is the same as it would be on a cube. If it is the case, the cuboid has genus 0, eight regions and the tiles of Y cannot be used in any terminal assembly. Otherwise, the cuboid has genus 1, less than eight regions, and at least one of the tile types of Y must be used in any terminal assembly. We next proceed with the formal proof.

Proof of Theorem 7.6. Let $C = C_0 \setminus C_0' \in O_1$ be an order-1 cuboid with the dimensions at least 10 for C_0' and α be an assembly of the TAS $\mathcal{S}_\mathcal{G} = (\Sigma, T, \sigma, str, \tau)$ such that its seed is placed at a normal placement. Note that if C_0 is too small there is no normal placement. According to Corollary 7.15 and Lemma 7.16, there is a tile type from $Y = \{t_{reg}\} \cup \{t_{mfs}\} \cup T_{ibc} \subseteq T$ in all terminal assemblies of $\mathcal{S}_\mathcal{G}$ on C if and only if there is tunnel on C (i.e. its genus is 1). □

7.8 Concluding remarks

We have shown that we can use the SFTAM to determine the genus of order-1 cuboids.

Despite the fact that our method is applicable only when the seed is placed on a normal placement, we do not consider it to be a strong restriction. Indeed, for large order-1 cuboids, unless one of the dimensions is exponentially small with respect to another one, almost all placements are normal placements. This creates a high probability for a seed

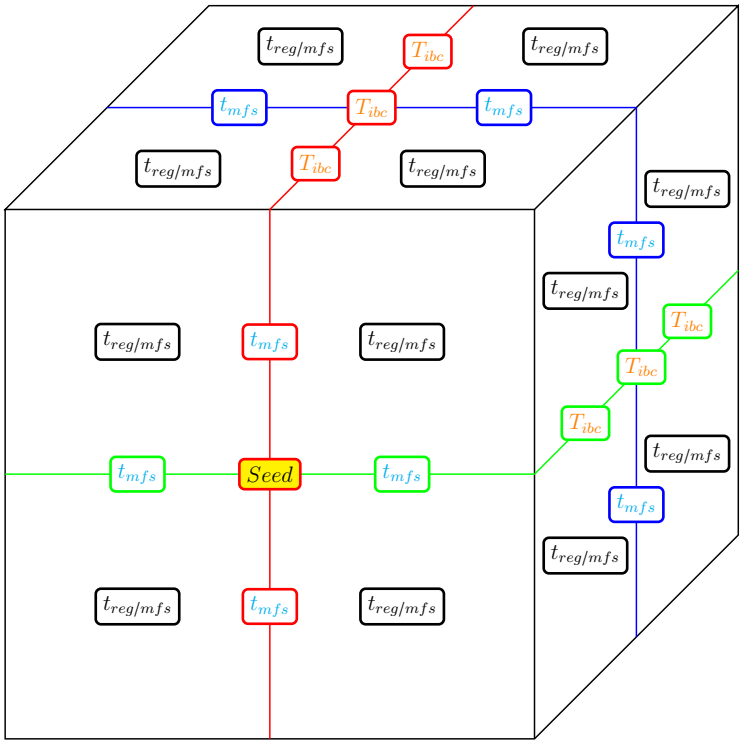


Figure 7.26: The places on a cuboid where, if there is a tunnel, a tile of Y must appear in the assembly.

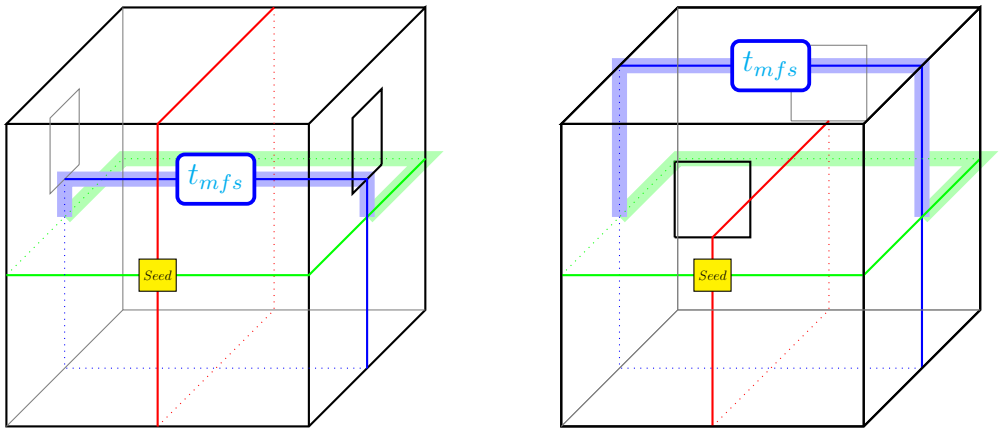
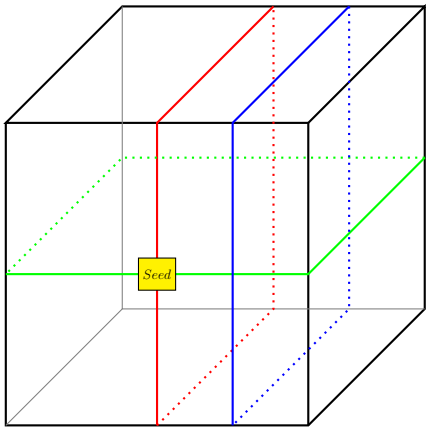


Figure 7.27: The closed ribbon formed by parts of the middle finding system (green) and the two ribbons of R_Z (blue), when two R_Z ribbons meet each other instead of reaching R_X. They meet the red ribbon R_X only once.

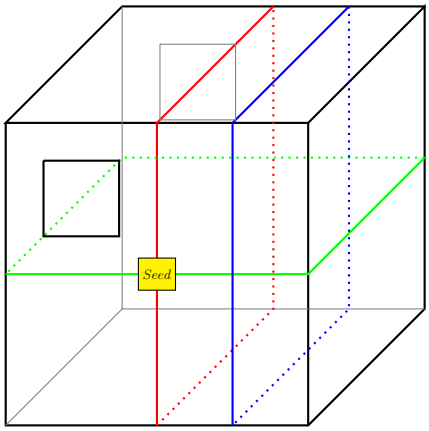
being placed in a normal placement, and hence having an assembly that detects the genus of the host of the order-1 cuboid.

The essential ingredient of the proof is the Middle Finding System, which enables to build a skeleton that partitions the cuboid into regions that can be distinguished. If we would apply a similar strategy without the Middle Finding System, the partition of the

surface of the cuboid into regions could fail, see for example Figure 7.28.



(a) Although there is no tunnel, the surface of the cuboid is partitioned into less than 8 regions.



(b) A tunnel is undetected because its two ends are in the same region.

Figure 7.28: A scenario where we apply the same strategy as in our work, but without using the Middle Finding System. Because the ribbons of R_Z do not start in the middle of R_Y, the regions do not partition the surface meaningfully.

The complexity of our assemblies illustrates the challenges of working on an unknown surface. For future work, it would be interesting to extend our results to a larger family of polycubes. In this work, the Middle Finding System was used to detect a potential tunnel on an order-1 cuboid. However, for more complicated surfaces, one needs to ensure that some part of the construction does go through the tunnel, and that it can be differentiated from the tiles it meets on the other side. Indeed, the idea of having regions with distinct identities can be reused in this context, but the Middle Finding System needs to be supplemented or replaced.

Chapter 8

Conclusion

In this thesis, we have studied the problem of determining the type of the surface on which an assembly is taking place, by using a specific set of tile types, whose tiles appear only for certain types of surfaces. We have first demonstrated this idea on flat surfaces to distinguish the flat torus, the horizontal flat cylinder, the vertical flat cylinder, and the infinite plane from each other. We then have given a more complicated solution for certain polycubes, in order to distinguish the ones of genus 0 and of genus 1.

To work on polycubes, we have introduced a new tile self-assembly model, the Surface Flexible Tile Assembly Model (SFTAM). We have shown that we can use self-assembly in this model to determine the genus of a given surface. For this, we have worked on a simple and special family of polycubes of genus at most 1, the order-1 cuboids. As our construction for order-1 cuboids is rather complex, it would be good to simplify it, in order to be able to generalize it more easily.

In both these results, it is interesting that we are able to determine a global property of the surface (the type of flat surface it belongs to, or the genus of the polycube) although tile self-assembly is a local process.

As possible future work, it would be interesting to extend our results to a larger family of polycubes. To do so, one possible first step would be to define the family O_n of order-n cuboids. We define O_n as the set of order-n cuboids for an integer n by a recursive definition: an order-n cuboid is obtained from the difference of an order-$(n-1)$-cuboid and an order-0 cuboid. Perhaps one can extend our main theorem by designing SFTAM self-assembly systems that can characterize order-n cuboids by their genus. However, the tools presented here are probably not sufficient. Some intermediate steps towards this goal could be tried first. For example, a simpler question is whether, for an order-n cuboid, we can distinguish the case that the genus is 0, from all other cases. A restricted class of order-n cuboids is that where all tunnels are placed on the same face. Perhaps our methods are applicable to that simpler problem on this simpler class of order-n cuboids.

We hope that one can do more with the SFTAM in the future. One could also use it to identify other properties of the host surface, such as the number of pits or concavities, or the dimensions of the surface.

Moreover, it seems natural to us to perform tile self-assembly on an existing arbitrary surface, and hence we believe that the SFTAM can be used in other contexts. One immediate possible extension of the SFTAM is to use it on surfaces other than polycubes. Indeed, if an orientable surface is quadrangulated (with only unit square faces), one can use the SFTAM. For example, one could use it on the surface of a quadrangulated "sheet

of paper", where bonds can be placed on the edges of the sheet of paper. More generally, the SFTAM could be used with quadrangulated surfaces whose quadrangular faces are not necessarily square. Indeed, there could be applications where the molecular tiles are stretchable and do not necessarily correspond to a unit square. For example, in Figure 8.1, an example from architecture shows a quadrangulated surface where the quadrangles have various shapes, that could be the host for such an extended SFTAM.

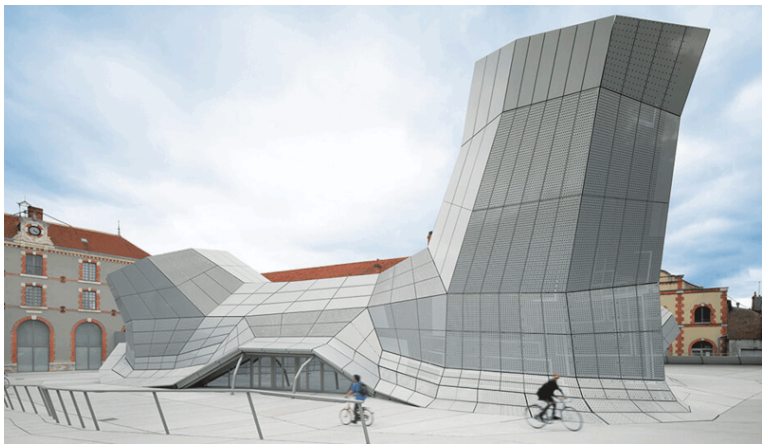


Figure 8.1: The surface of the building of the FRAC (Fonds Régional d'Art Contemporain) in Orléans is a quadrangulation, so one could try to perform tile self-assembly on it.[1]

Another possible extension of the SFTAM would be an orientable version of the SFTAM, if we wish to work on a surface that is not necessarily orientable, such as a Möbius strip. Here, the main difference would be how a placement of a tile can be described. Indeed, a non-orientable surface may not have the notion of interior and exterior, and one would need to also associate a *normal vector* to a placement of a tile, to show how the tile must be placed. The idea of a normal vector was already used in the FTAM (Section 3.5.1), since in that context, the tiles are placed in 3D space, so they needed to describe more precisely the orientation of the tile. This idea could be used in the context of general surfaces as well.

Furthermore, it would be very interesting to know if there is a difference between the computations that can be performed using self-assembly on surfaces of different genus.

As tile self-assembly has important practical applications and can be performed on many types of surfaces/objects, we hope that this thesis will inspire more work in this research direction, with more studies on the genus of the host surfaces. As a setting relevant to practical applications, one could imagine to have 3D objects that need to be identified. One could coat them using SFTAM assemblies that contain some special marking tiles: the configuration of these marking tiles would help for this task of identification. This setting resembles our work, where the marking tiles would be the ones of the set Y. This is a sort of computational coating.

Generally, as self-assembly is used in applications as coating of surfaces (for example for coating drug molecules), perhaps our work can be used in this kind of setting. Another potential application could be to detect certain diseases: for example, in certain diseases, like the sickle cell disease, the red blood cells of affected patients have a different shape than in healthy individuals. One could thus use similar assemblies as the ones from this thesis to detect the disease.

[1] Image from `https://www.azuremagazine.com/article/the-fantastic-frac-centre-in-france/`.

Publications obtained during the PhD

Chapters 5, 6 and 7 are expanded from a paper presented at the DNA 29 conference in September 2023 and published as [14].

In the beginning of my PhD, I collaborated on a separate project that was published in [13] but which is not included in this thesis.

Bibliography

[1] Z. Abel, N. Benbernou, M. Damian, E. D. Demaine, M. L. Demaine, R. Flatland, S. Kominers and R. Schweller. Shape replication through self-assembly and RNase enzymes. Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), pages 1045–1064, 2010.

[2] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science* 266(5187):1021–1024, 1994.

[3] L. Adleman, Q. Cheng, A. Goel and M. Huang. Running time and program size for self-assembled squares. Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001), pages 740–748, 2001.

[4] L. Adleman, Q. Cheng, A. Goel, M. Huang, D. Kempe, P. Moisset de Espanès and P. W. K. Rothemund. Combinatorial optimization problems in self-assembly. Proceedings of the 34th annual ACM symposium on Theory of computing (STOC 2002), pages 23–32, 2002.

[5] O. Aichholzer, M. Biro, E. D. Demaine, M. L. Demaine, D. Eppstein, S. P. Fekete, A. Hesterberg, I. Kostitsyna and C. Schmidt. Folding polyominoes into (poly)cubes. *International Journal of Computational Geometry and Applications* 28(3):197–226, 2018.

[6] G. Aloupis, P. Bose, S. Collette, E. D. Demaine, M. L. Demaine, L. Douïeb, V. Dujmović, J. Iacono, S. Langerman and P. Morin. Common unfoldings of polyominoes and polycubes. Proceedings of the International Conference on Computational Geometry, Graphs and Applications (CGGA 2010), *Lecture Notes in Computer Science* 7033:44–54, 2010.

[7] A. Alseth, D. Hader and M. J. Patitz. Universal shape replication via self-assembly with signal-passing tiles. Proceedings of the 28th International Conference on DNA Computing and Molecular Programming (DNA 28), *LIPIcs* 238:2:1–2:24, 2022.

[8] A. Alseth, J. Hendricks, M. J. Patitz and T. A. Rogers. Replication of arbitrary hole-free shapes via self-assembly with signal-passing tiles. *New Generation Computing* 40:553–601, 2022.

[9] M. A. Armstrong. *Basic topology.* Undergraduate Texts in Mathematics, 1983.

[10] R. D. Barish, R. Schulman, P. W. K. Rothemund and E. Winfree. An information-bearing seed for nucleating algorithmic self-assembly. *Proceedings of the National Academy of Sciences* 106(15):6054–6059, 2009.

[11] A. T. Becker, S. P. Fekete, P. Keldenich, D. Krupke, C. Rieck, C. Scheffer and A. Schmidt. Tilt assembly: Algorithms for micro-factories that build objects with uniform external forces. *Algorithmica* 82(2):165-187, 2020.

[12] F. Becker. *Géométrie pour l'auto-assemblage*. PhD thesis, ENS de Lyon, 2008.

[13] F. Becker, T. Besson, J. Durand-lose, A. Emmanuel, M. Foroughmand-Araabi, S. Goliaei and S. Heydarshahi. Abstract Geometrical Computation 10: An Intrinsically Universal Family of Signal Machines. *ACM Transactions on Computation Theory* 13(1):4, 2021.

[14] F. Becker and S. Heydarshahi. DNA tile self-assembly for 3D-surfaces: Towards genus identification. Proceedings of the 29th International Conference on DNA Computing and Molecular Programming (DNA 29), *LIPIcs* 276, 8:1–8:20, 2023.

[15] F. Becker, I. Rapaport and E. Rémila. Self-assemblying classes of shapes with a minimum number of tiles, and in optimal time. Proceedings of the 26th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2006), *Lecture Notes in Computer Science* 4337:45–56, 2006.

[16] F. Becker, E. Rémila and N. Schabanel. Time optimal self-assembly for 2D and 3D shapes: the case of squares and cubes. Proceedings of the 14th International Conference on DNA Computing and Molecular Programming (DNA 14), *Lecture Notes in Computer Science* 5347:144–155, 2008.

[17] R. Berger. Undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 1965.

[18] J. Bohlin. *Design and modular self–assembly of nanostructures*. PhD thesis, University of Oxford, 2022.

[19] J. Bohlin, A. J. Turberfield, A. A. Louis and P. Šulc. Designing the self-assembly of arbitrary shapes using minimal complexity building blocks. *ACS Nano* 17(6):5387–5398, 2023.

[20] H. R. Brahana. Systems of circuits on two-dimensional manifolds. *Annals of Mathematics* 23(2):144–168, 1921.

[21] C. J. Brinker, Y. Lu, A. Sellinger. and H. Fan. Evaporation-induced self-assembly: nanostructures made easy. *Advanced Materials* 11:579–585, 1999.

[22] C. T. Chalk, E. D. Demaine, M. L. Demaine, E. Martinez, R. T. Schweller, L. Vega and T. Wylie. Universal shape replicators via self-Assembly with attractive and repulsive forces. Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017), pages 225–238, 2017.

[23] C. C. Chang and C. Y. Lin. Texture tiling on 3D models using automatic polycube-maps and Wang tiles. *Journal of Information Science & Engineering* 26(1):291–305, 2010.

[24] H.-L. Chen and A. Goel. Error free self-assembly using error prone tiles. Proceedings of the 10th International Conference on DNA Computing and Molecular Programming (DNA 10), *Lecture Notes in Computer Science* 3384:62–75, 2004.

[25] I. R. Cohen and A. Marron. The evolution of universal adaptations of life is driven by universal properties of matter: energy, entropy, and interaction. *F1000Research* 9:626, 2020.

[26] M. Cook, Y. Fu and R. Schweller. Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011), 2011.

[27] M. Cook, T. Stérin and D. Woods. Small tile sets that compute while solving mazes. Proceedings of the 27th International Conference on DNA Computing and Molecular Programming (DNA 27), *LIPIcs* 205, 8:1–8:20, 2021.

[28] M. da Ronch, M. Gandit and S. Gravier. Du problème de Wang vers une nouvelle situation de recherche pour la classe. *Repères IREM* 121:77–105, 2020.

[29] D. Doty, J. H. Lutz, M. J. Patitz, . T. Schweller, S. M. Summers and D. Woods. The tile assembly model is intrinsically universal. Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012), pages 302–310, 2012.

Intrinsic universality in self-assembly. Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS 2010), *LIPIcs* 5:275–286, 2010.

[30] J. Durand-Lose, J. Hendricks, M. J. Patitz, I. Perkins and M. Sharp. Self-assembly of 3-D structures using 2-D folding tiles. *Natural Computing* 19, 337–355, 2020.

[31] L. Euler. Elementa doctrinae solidorum. *Novi Commentarii academiae scientiarum Petropolitanae* 4, 109–140, 1758. Republished in *Opera Omnia, Serie 1* 26:71–93, 1992.

[32] C. G. Evans. *Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly.* PhD thesis, California Institute of Technology, Los Angeles, 2014.

[33] D. Furcy and S. M. Summers. Optimal self-assembly of finite shapes at temperature 1 in 3D. *Algorithmica* 80:1909–1963, 2018.

[34] J. Gallier and D. Xu. *A guide to the Classification Theorem for Compact Surfaces.* Springer, 2013.

[35] C. Geary, P.-É. Meunier, N. Schabanel and S. Seki. Oritatami: a computational model for molecular co-transcriptional folding. *International Journal of Molecular Sciences* 20(9):2259, 2019.

[36] C. Geary, P.-É. Meunier, N. Schabanel and S. Seki. Proving the Turing universality of oritatami cotranscriptional folding. Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC 2018), *LIPIcs* 23:1–23:13, 2018.

[37] M. Godin, V. Tabard-Cossa, Y. Miyahara, T. Monga, P. J. Williams, L. Y. Beaulieu, R. B. Lennox and P. Grutter. Cantilever-based sensing: The origin of surface stress and optimization strategies. *Nanotechnology* 21(7):075501, 2010.

[38] D. Han, S. Pal, J. Nangreave, Z. Deng, Y. Liu and H. Yan. DNA origami with complex curvatures in three-dimensional space. *Science* 332(6027):342–346, 2011.

[39] F. Harary. *Graph Theory*. Addison-Wesley, 1994.

[40] J. Hendricks. *Simulation in algorithmic self-assembly*. PhD thesis, University of Arkansas, Fayetteville, 2015.

[41] E. Jeandel and M. Rao. An aperiodic set of 11 Wang tiles. *Advances in Combinatorics* 2021.

[42] M. B. Jones, N. C. Seeman and C. A. Mirkin. Programmable Materials and the Nature of the DNA Bond. *Science* 347:840–840, 2015.

[43] M. Y. Kao and V. Ramachandran. DNA self-assembly for constructing 3D boxes. Proceedings of the 12th International Symposium on Algorithms and Computation (ISAAC 2001), *Lecture Notes in Computer Science* 2223:429–441, 2001.

[44] A. Keenan, R. Schweller and X. Zhong. Exponential replication of patterns in the signal tile assembly model. *Natural Computing* 14(2):265–278, 2015.

[45] J. Keller, C. Rieck, C. Scheffer and A. Schmidt. Particle-based assembly using precise global control. *Algorithmica* 84:2871–2897, 2022.

[46] J. I. Lathrop, J. H. Lutz, M. J. Patitz and S. M. Summers. Computability and complexity in self-assembly. *Theory of Computing Systems* 48(3):617–647, 2011.

[47] K. Lindgren and M. Nordahl. Universal computation in simple one-dimensional cellular automata. *Complex Systems* 4(3):299–318, 1990.

[48] D. Liu, M. Wang, Z. Deng, R. Walulu and C. Mao. Tensegrity: construction of rigid DNA triangles with flexible four-arm DNA junctions. *Journal of the American Chemistry Society* 126(8):2324–2325, 2004.

[49] W. Liu, H. Zhong, R. Wang and N. C. Seeman. Crystalline two-dimensional DNA-origami arrays. *Angewandte Chemie International Edition* 50(1):264–267, 2011.

[50] B. Lu, S. Vecchioni, Y. P. Ohayon, J. W. Canary and R. Sha. The wending rhombus: Self-assembling 3D DNA crystals. *Biophysical Journal* 121(24):4759–4765, 2022.

[51] D. Mackenzie. The Poincaré Conjecture – Proved. *Science* 314:1848–1849, 2006.

[52] P.-É. Meunier, D. Regnault and D. Woods. The program-size complexity of self-assembled paths. Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020), pages 727–737, 2020.

[53] P.-É. Meunier and D. Regnault. Directed non-cooperative tile assembly is decidable. Proceedings of the 27th International Conference on DNA Computing and Molecular Programming (DNA 27), *LIPIcs* 205, 6:1–6:21, 2021.

[54] P.-É. Meunier and D. Woods. The non-cooperative tile assembly model is not intrinsically universal or capable of bounded Turing machine simulation. Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017), pages 328–341, 2017.

[55] G. Miller and D. Knuth. *Cubigami.* `http://www.puzzlepalace.com/#/puzzles/200610`

[56] V. Muñoz, Á. González-Prieto and J. Á. Rojo. *Geometry and topology of manifolds: surfaces and beyond.* Series: Graduate Studies in Mathematics 208, 2020.

[57] M. J. Patitz. *Python-based Tile Assembly Simulator (PyTAS).* `http://self-assembly.net/software/PyTAS/`.

[58] M. J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing* 13, 195–224, 2014.

[59] M. J. Patitz and S. M. Summers. Self-assembly of infinite structures: A survey. *Theoretical Computer Science* 412:159–165, 2011.

[60] M. J. Patitz and S. M. Summers. Identifying shapes using self-assembly. *Algorithmica* 64:481–510, 2012.

[61] D. Pchelina, N. Schabanel, S. Seki and G. Theyssier. Oritatami systems assemble shapes no less complex than tile assembly model (ATAM). Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022), *LIPIcs*: 51:1-51:23, 2022.

[62] P. Popescu-Pampu. *What is the genus?* Lecture Notes in Mathematics 2162, *Springer*, 2016.

[63] P. W. K. Rothemund. *Theory and experiments in algorithmic self-assembly.* Ph.D. thesis, University of Southern California, Los Angeles, 2001.

[64] P. W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature* 440(7082):297–302, 2006.

[65] P. W. K. Rothemund and E. Winfree. The program-size complexity of self-assembled squares. Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000), pages 459–468, 2000.

[66] N. C. Seeman. Nucleic acid junctions and lattices. *Journal of Theoretical Biology* 99(2):237–247, 1982.

[67] DNA Nanotechnology: Bibliography from Ned Seeman's Laboratory. `http://seemanlab4.chem.nyu.edu/nanobib.html`

[68] V. A. Sontakke and Y. Yokobayashi. Programmable Macroscopic Self-Assembly of DNA-Decorated Hydrogels. *Journal of the American Chemical Society* 144 (5), 2149-2155, 2022.

[69] Q. Tang and D. Han. Obtaining Precise Molecular Information via DNA Nanotechnology. *Membranes* 11(9):683, 2021.

[70] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 2-42(1):230–65, 1937.

[71] M. M. de Villiers, D. P. Otto, S. J. Strydom and Y. M. Lvov. Introduction to nanocoatings produced by layer-by-layer (LbL) self-assembly. *Advanced Drug Delivery Reviews* 63(9):701–715, 2011.

[72] H. Wang. Proving theorems by pattern recognition – II. *Bell System Technical Journal* 40 (1):1–41, 1961.

[73] J. D. Watson and F. H. C. Crick. A structure for deoxyribose nucleic acid. *Nature* 171:737–738, 1953.

[74] G. M. Whitesides, J. K. Kriebel and J. C. Love. Molecular engineering of surfaces using self-assembled monolayers. *Science Progress* 88(1):17–48, 2005.

[75] E. Winfree. *Algorithmic Self-Assembly of DNA.* Ph.D.thesis, California Institute of Technology, 1998.

[76] E. Winfree. Simulations of computing by self-assembly. *DNA Based Computers IV*, 1998.

[77] E. Winfree, X. Yang and N. C. Seeman. Universal computation via self-assembly of DNA: Some theory and experiments. In *DNA Based Computers II*, volume 44 of DIMACS, pages 191–213. American Mathematical Society, 1996.

[78] D. Woods, D. Doty, C. Myhrvold, J. Hui, F. Zhou, P. Yin and E. Winfree. Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. *Nature* 567(7748):366–372, 2019.

[79] J. F. Woods, L. Gallego, P. Pfister, M. Maaloum, A. V. Jentzsch and M. Rickhaus. Shape-assisted self-assembly. *Nature Communications* 13:3681, 2022.

[80] Y. Yang, J. Wang, H. Shigematsu, W. Xu, W. M. Shih, J. E. Rothman and C. Lin. Self-assembly of size-controlled liposomes on DNA nanotemplates. *Nature Chemistry* 8(5):476–483, 2016.

[81] Y. Yu, L. Chen, D. Weng, J. Wang, C. Chen and A. Mahmood. A promising self-assembly PTFE coating for effective large-scale deicing. *Progress in Organic Coatings* 147:105732, 2020.

[82] F. Zhang, C. R. Simmons, J. Gates, Y. Liu and H. Yan. Self-assembly of a 3D DNA crystal structure with rationally designed six-fold symmetry. *Angewandte Chemie International Edition* 57(38):12504-12507, 2018.

[83] S. Zhung. Fabrication of novel biomaterials through molecular self-assembly. *Nature Biotechnology* 21:1171–1178, 2003.

[84] R. Zhuo, F. Zhou, X. He, R. Sha, N. C. Seeman and P. M. Chaikin. Litters of self-replicating origami cross-tiles. *Biophysics and Computational Biology* 116(6):1952–1957, 2019.

List of Figures

List of Tables

Shahrzad HEYDARSHAHI

Auto-assemblage sur surfaces diverses

Résumé : Nous étudions l'auto-assemblage par tuiles sur divers types de surfaces. L'auto-assemblage par tuiles est un processus dynamique basé sur des tuiles carrées ayant quatre côtés distingués, qui peuvent s'attacher deux à deux sous certaines conditions. Le processus commence avec une configuration initiale appelée graine, et s'arrête si aucune autre tuile ne peut être ajoutée. Ce modèle peut être implémenté en pratique via les nanotechnologies de l'ADN, et il est étudié depuis la fin des années 1990. Il a été démontré que ce modèle permet de réaliser des calculs arbitraires : il s'agit d'un modèle de calcul universel. Nous étudions l'auto-assemblage par tuiles d'un point de vue théorique. La plupart des travaux sur ce sujet portent sur l'auto-assemblage par tuiles sur le plan Euclidien 2D. Nous nous intéressons à son comportement sur des surfaces plus complexes. La question centrale que nous posons est si l'on peut concevoir un système d'auto-assemblage par tuiles permettant de détecter le type de surface sur laquelle il est effectué ? Nous nous intéressons d'abord au cas de quatre types de surfaces plates : le tore plat, le cylindre vertical plat, le cylindre horizontal plat, et le plan Euclidien. Nous présentons un système d'auto-assemblage par tuiles dans le modèle classique abstract Tile Assembly Model (aTAM). Notre système peut être appliqué sur toute surface appartenant à l'un des quatre types de surfaces, et présente des particularités uniques en fonction du type de surface. Plus précisément, certaines tuiles apparaissent uniquement si l'assemblage a lieu sur l'un de ces types de surfaces. Nous abordons également des surfaces 3D plus complexes : celles d'objets 3D appelés polycubes. Les polycubes sont constitués de cubes unitaires collés les uns aux autres via leurs faces. La propriété la plus fondamentale d'une surfaces est sans doute son genre ; pour simplifier, il s'agit du nombre de trous qui perçent la surface. Notre but est de créer un système d'auto-assemblage par tuiles capable de détecter le genre du polycube sous-jacent. Afin d'effectuer de l'auto-assemblage par tuiles sur la surface d'un polycube, nous définissons tout d'abord un nouveau modèle adapté, appelé Surface Flexible Tile Assembly Model (SFTAM). Ce modèle étend le modèle aTAM et est inspiré par un autre modèle, le Flexible Tile Assembly Model (FTAM). Nous conçevons un système d'auto-assemblage par tuiles qui détecte le genre d'un type particulier de polycubes. Ces polycubes sont appelés cuboïdes d'ordre 1, et ils peuvent avoir genre 0 ou genre 1. Notre système est tel que, dans tout assemblage terminal, si le genre est 1, une tuile d'un ensemble particulier Y de types de tuiles apparaît nécessairement. Si le genre est 0, les tuiles dont le type est dans Y n'apparaissent dans aucun assemblage productible.

Mots clés : Auto-assemblage par tuiles, Calcul ADN, Surfaces géométriques, Genre, Polycubes

Self-Assembly on Various Surfaces

Abstract : We investigate tile self-assembly on various types of surfaces. Tile self-assembly is a dynamic process based on square tiles that have four distinguished sides that can attach to each other under certain conditions. The process starts with a specific predetermined configuration called a seed, and stops if no further tile can be attached. This model can be implemented in practice using DNA nanotechnology and has been investigated since the late 1990s. It has been shown theoretically to allow performing arbitrary computations : it is a universal computation model. We study tile self-assembly from the theoretical side. Most works in the area deal with the Euclidean 2D plane. We are interested in studying its behaviour on more complex surfaces. The central question that we ask, is, can we design a tile self-assembly system that can detect the type of surface that it is performed on? We first address this question for the case of four types of flat surfaces: the flat torus, the flat vertical cylinder, the flat horizontal cylinder, and the Euclidean plane. We design a tile self-assembly system in the classic abstract Tile Assembly Model (aTAM) that can be performed on any surface from one of these four types, and that exhibits specific features for each type. More precisely, certain tiles will uniquely appear if the assembly is taking place on one of these types of surfaces. We are also interested in more complex 3D surfaces that form the surface of objects called polycubes. Polycubes are made from unit cubes that are glued to each other by their faces. Perhaps the most fundamental property of a surface is its genus; informally, it is the number of holes piercing the surface. Our goal is to design a tile self-assembly system that can detect the genus of the underlying polycube surface. In order to perform tile self-assembly on polycubes, we first define a new suitable model, that we call the Surface Flexible Tile Assembly Model (SFTAM). This model extends the aTAM and is inspired by another existing model, the Flexible Tile Assembly Model (FTAM). We design a tile self-assembly system that detects the genus of a special type of polycubes. These polycubes are called order-1 cuboids, and they can have genus 0 or genus 1. In any terminal assembly of our system, if the genus is 1, a tile from a special subset Y of tile types must appear. When the genus is 0, however, tiles of Y never appear in any producible assembly.

Keywords : Tile self-assembly, DNA computing, Geometric surfaces, Genus, Polycubes

LABORATOIRE D'INFORMATIQUE FONDAMENTALE D'ORLEANS

Laboratoire d'Informatique Fondamentale d'Orléans
6 rue Léonard de Vinci, 45100 Orléans, France