Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 395 (2008) 3–23

Theoretical Computer Science

www.elsevier.com/locate/tcs

Nondeterministic polynomial time factoring in the tile assembly model

Yuriy Brun[*]

Department of Computer Science, University of Southern California, Los Angeles, CA 90089, United States

Received 21 November 2006; received in revised form 26 July 2007; accepted 30 July 2007

Communicated by L. Kari

Abstract

Formalized study of self-assembly has led to the definition of the tile assembly model, Previously I presented ways to compute arithmetic functions, such as addition and multiplication, in the tile assembly model: a highly distributed parallel model of computation that may be implemented using molecules or a large computer network such as the Internet. Here, I present tile assembly model systems that factor numbers nondeterministically using $\Theta(1)$ distinct components. The computation takes advantage of nondeterminism, but theoretically, each of the nondeterministic paths is executed in parallel, yielding the solution in time linear in the size of the input, with high probability. I describe mechanisms for finding the successful solutions among the many parallel executions and explore bounds on the probability of such a nondeterministic system succeeding and prove that the probability can be made arbitrarily close to 1.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Self-assembly; Factoring; Tile assembly model; Crystal growth; Molecular computation; Natural computation; Distributed computing; Parallel computing; Nondeterministic computation

1. Introduction

Self-assembly is a process that is ubiquitous in nature. Systems form on all scales via self-assembly: atoms self-assemble to form molecules, molecules to form complexes, and stars and planets to form galaxies. One manifestation of self-assembly is crystal growth: molecules self-assembling to form crystals. Crystal growth is an interesting area of research for computer scientists because it has been shown that, in theory, under careful control, crystals can compute [39]. The field of DNA computation demonstrated that DNA can be used to compute [1], solving NP-complete problems such as the satisfiability problem [12,11]. This idea of using molecules to compute nondeterministically is the driving motivation behind my work.

Winfree showed that DNA computation is Turing suniversal [38]. While DNA computation suffers from relatively high error rates, the study of self-assembly shows how to utilize redundancy to design systems with built-in error

[*] Tel.: +1 3023545174.
E-mail address: ybrun@usc.edu.

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.07.051

correction [42,10,21,27,41]. Researchers have used DNA to assemble crystals with patterns of binary counters [8] and Sierpinski triangles [33], but while those crystals are deterministic, generating nondeterministic crystals may hold the power to solving complex problems quickly.

Two important questions about self-assembling systems that create shapes or compute functions are: "what is a minimal tile set that can accomplish this goal?" and "what is the minimum assembly time for this system?" Here, I study systems that factor numbers and ask these questions, as well as another that is important to nondeterministic computation: "what is the probability of assembling the crystal that encodes the solution?"

Adleman has emphasized studying the number of steps it takes for an assembly to complete (assuming maximum parallelism) and the minimal number of tiles necessary to assemble a shape [2]. He answered these questions for n-long linear polymers [3]. Previously, I have extended these questions to apply to systems that compute functions, rather than assemble shapes [13], and now I extend them to systems that compute functions nondeterministically.

Adleman proposed studying the complexity of tile systems that can uniquely produce $n \times n$ squares. A series of researchers [34,4,5,23] proceeded to answer the questions: "what is a minimal tile set that can assemble such shapes?" and "what is the assembly time for these systems?" They showed that, for most n, the minimal tile set that assembles $n \times n$ squares is of size $\Theta(\frac{\log n}{\log \log n})$ and the optimal assembly time is $\Theta(n)$ [5]. A key issue related to assembling squares is the assembly of small binary counters, which theoretically can have as few as seven tile types [23].

Soloveichik et al. studied assembling all decidable shapes in the tile assembly model and found that the minimal set of tiles necessary to uniquely assemble a shape is directly related to the Kolmogorov complexity of that shape. Interestingly, they found that for the result to hold, scale must not be a factor. That is, the minimal set of tiles they find builds a given shape (e.g. square, a particular approximation of the map of the world, etc.) on some scale, but not on all scales. Thus they showed that smaller versions of the same shape might require larger sets of tiles to assemble [36].

I proposed and studied systems that compute the sums and products of two numbers using the tile assembly model [13]. I found that in the tile assembly model, adding and multiplying can be done using $\Theta(1)$ tiles (as few as eight tiles for addition and as few as 28 tiles for multiplication), and that both computations can be carried out in time linear in the input size. I also speculated that those systems could be modified to work in sequence with other systems, in effect chaining computations together. In this paper I use just that technique: I utilize the multiplication system as a subroutine for the factoring system, exploring nondeterministic computation in the tile assembly model.

Other early attempts at nondeterministic computation include a proposal by Lagoudakis et al. to solve the satisfiability problem [26]. They informally define a system that nondeterministically computes whether or not an n-variable boolean formula is satisfiable using $\Theta(n^2)$ distinct tiles. In contrast, the system I present in this paper for solving an NP-complete problem uses $\Theta(1)$ distinct tiles and assembles in time linear in the input.

While the constructions in this paper are in some ways analogous to traditional computer programs, and their running times are polynomially related to the running times of Turing machines and nondeterministic Turing machines, Baryshnikov et al. began the study of fundamental limits on the time required for a self-assembly system to compute functions [9]. They consider models of molecular self-assembly and apply Markov models to show lower limits on assembly times.

Researchers have also studied variations of the traditional tile assembly model. Aggarwal et al. and Kao et al. have shown that changing the temperature of assembly from a constant throughout the assembly process to a discrete function reduces the minimal tile set that can build an $n \times n$ square to a size $\Theta(1)$ tile set [7,25].

Barish et al. have demonstrated DNA implementations of tile systems, one that copies an input and another that counts in binary [8]. Similarly, Rothemund et al. have demonstrated a DNA implementation of a tile system that computes the *xor* function, resulting in a Sierpinski triangle [33]. These systems grow crystals using double-crossover complexes [24] as tiles. The theoretical underpinnings of these systems are closely related to the work presented here because these systems compute functions.

Rothemund has demonstrated what is currently the state-of-the-art of DNA nanostructure design and implementation with DNA origami, a concept of folding a single long scaffold strand into an arbitrary shape by using small helper strands [31,30,32]. Similar concepts may be the key to three-dimensional self-assembly, more powerful error-correction techniques, and self-assembly using biological molecules.

Cook et al. have explored using the tile assembly model to implement arbitrary circuits [22]. Their model allows for tiles that contain gates, counters, and even more complex logic components, as opposed to the simple static tiles used in the traditional tile assembly model and in this paper. While they speculate that the tile assembly model logic may be used to assemble logic components attached to DNA, my assemblies require no additional logic components and encode the computation themselves. It is likely that their approach will require fewer tile types and perhaps assemble faster, but at the disadvantage of having to not only assemble crystals but also attach components to those crystals and create connections among those components. Nevertheless, Rothemund's work with using DNA as a scaffold may be useful in attaching and assembling such components [32].

Some experimental work [1,11] has shown that it is possible to work with an exponential number of components and even to solve NP-complete problems. I explore the possibility of nondeterministic computation using the tile assembly model and prove bounds on the probability of successful computation. The probability of successfully factoring a number can be made arbitrarily close to 1 by increasing the number of self-assembling components and seeds in the computation.

A preliminary version of the constructions presented in this paper has appeared in [14], though without the formal proofs and analysis presented here.

The rest of this paper is structured as follows: Section 1.1 will describe in detail the tile assembly model, Section 2 will discuss what it means for a tile assembly model system to compute functions deterministically and nondeterministically, Section 3 will introduce, define, and prove the correctness of two factoring systems, and Section 4 will summarize the contributions of this work.

1.1. Tile assembly model

The tile assembly model [40,39,34] is a formal model of crystal growth. It was designed to model self-assembly of molecules such as DNA. It is an extension of a model proposed by Wang [37]. The model was fully defined by Rothemund and Winfree [34], and the definitions here are similar to those, and identical to the ones in [13], but I restate them here for completeness and to assist the reader. Intuitively, the model has *tiles* or squares that stick or do not stick together based on various binding domains on their four sides. Each tile has a binding domain on its north, east, south, and west side, and may stick to another tile when the binding domains on the abutting sides of those tiles match and the total strength of all the binding domains on that tile exceeds the current temperature. The four binding domains define the type of the tile. While this definition does not allow tiles to rotate, it is essentially equivalent to a system with rotating tiles.

Formally, let Σ be a finite alphabet of binding domains such that $null \in \Sigma$. I will always assume $null \in \Sigma$ even when I do not specify so explicitly. A **tile** over a set of binding domains Σ is a 4-tuple $\langle \sigma_N, \sigma_E, \sigma_S, \sigma_W \rangle \in \Sigma^4$. A **position** is an element of $\mathbb{Z}^2$. The set of directions $D = \{N, E, S, W\}$ is a set of four functions from positions to positions, i.e. $\mathbb{Z}^2$ to $\mathbb{Z}^2$, such that for all positions (x, y), $N(x, y) = (x, y + 1)$, $E(x, y) = (x + 1, y)$, $S(x, y) = (x, y - 1)$, $W(x, y) = (x - 1, y)$. The positions (x, y) and (x', y') are neighbors iff $\exists d \in D$ such that $d(x, y) = (x', y')$. For a tile t, for $d \in D$, I will refer to $bd_d(t)$ as the binding domain of tile t on d's side. A special tile $empty = \langle null, null, null, null \rangle$ represents the absence of all other tiles.

A **strength function** $g : \Sigma \times \Sigma \to \mathbb{N}$, where g is commutative and $\forall \sigma \in \Sigma\ g(null, \sigma) = 0$, denotes the strength of the binding domains. It is common to assume that $g(\sigma, \sigma') = 0 \iff \sigma \neq \sigma'$. This simplification of the model implies that the abutting binding domains of two tiles have to match to bind.

Let T be a set of tiles containing the empty tile. A **configuration** of T is a function $A : \mathbb{Z} \times \mathbb{Z} \to T$. I write $(x, y) \in A$ iff $A(x, y) \neq empty$. A is finite iff there is only a finite number of distinct positions $(x, y) \in A$.

Finally, a **tile system** $\mathbb{S}$ is a triple $\langle T, g, \tau \rangle$, where T is a finite set of tiles containing $empty$, g is a strength function, and $\tau \in \mathbb{N}$ is the temperature.

If A is a configuration, then within system $\mathbb{S}$, a tile t can **attach** to A at position (x, y) and produce a new configuration A' iff:

- $(x, y) \notin A$, and
- $\sum_{d \in D} g(bd_d(t), bd_{d^{-1}}(A(d(x, y)))) \geq \tau$, and
- $\forall (u, v) \in \mathbb{Z}^2, (u, v) \neq (x, y) \Rightarrow A'(u, v) = A(u, v)$, and
- $A'(x, y) = t$.

That is, a tile can attach to a configuration only in empty positions and only if the total strength of the appropriate binding domains on the tiles in neighboring positions meets or exceeds the temperature τ. For example, if for all σ, $g(\sigma, \sigma) = 1$ and $\tau = 2$ then a tile t can attach only at positions with matching binding domains on the tiles in at least two adjacent positions.

Given a tile system $\mathbb{S} = \langle T, g, \tau \rangle$, a set of tiles Γ, and a **seed configuration** $S : \mathbb{Z}^2 \to \Gamma$, if the above conditions are satisfied, one may attach tiles of T to S. Configurations produced by repeated attachments of tiles from T are said to be **produced** by $\mathbb{S}$ on S. If this process terminates, then the configuration achieved when no more attachments are possible is called the **final configuration**. At certain times, it may be possible for more than one tile to attach at a given position, or there may be more than one position where a tile can attach. If for all sequences of tile attachments, all possible final configurations are identical, then $\mathbb{S}$ is said to produce a **unique** final configuration on S.

Note that a system may produce a unique final configuration, even though there exist non-unique sequences of attachments that continue growing at infinitum. Theoretically, such constructions pose no problem, though they may present problems to certain implementations of tile systems. In particular, the infinite configurations might consume all the tiles available for construction. It is possible to limit the definition of a unique final configuration to exclude systems that produce infinite configurations; however, such a restriction seems somewhat arbitrary and would only be helpful for some implementations of the tile assembly model. Further, only some systems that assemble infinite configurations would suffer from those configurations consuming a majority of material. In particular, the factoring systems presented in this paper allow for the assembly of infinite configurations, but the probability of a configuration assembling is inversely proportional to its size, so very large configurations are unlikely to form, and a physical implementation of such a system would use up a small fraction of the material on the large configurations.

I choose not to restrict my definitions here, though an alternate set of definitions that takes the probabilities of configurations forming into account is possible. The systems presented in this paper would satisfy a stricter definition that concerned itself with making sure a large portion of the tiles formed small configurations.

Let $\mathbb{S} = \langle T, g, \tau \rangle$, and let S_0 be a seed configuration such that $\mathbb{S}$ produces a unique final configuration F on S_0. Let $W_0 \subseteq 2^{T \times \mathbb{Z}^2}$ be the set of all tile–position pairs $\langle t, (x, y) \rangle$ such that t can attach to S_0 at (x, y). Let S_1 be the configuration produced by adding all the elements of W_0 to S_0 in one time step. Define $W_1, W_2, \ldots$ and $S_2, S_3, \ldots$ similarly. Let n be the smallest natural number such that $S_n \equiv F$. Then n is the **assembly time** of $\mathbb{S}$ on S_0 to produce F.

I allow the codomain of S to be Γ, a set of tiles which may be different from T. The reason is that I will study systems that compute functions using minimal sets T; but the seed, which has to code for the input of the function, may contain more distinct tiles than there are in T. Therefore, I wish to keep the two sets separate. Note that, at any temperature τ, it takes $\Theta(n)$ distinct tiles to assemble an arbitrary n-bit input such that each tile codes for exactly one of the bits.

Winfree showed that the tile assembly model with $\tau = 2$ is Turing universal [39] by showing that a tile system can simulate Wang tiles [37], which Robinson showed to be universal [29]. Adleman et al. showed that the tile assembly model with $\tau = 1$ is Turing universal [6].

2. Computing functions

Let $\mathbb{N} = \mathbb{Z}_{\geq 0}$. Intuitively, in order for a tile system to deterministically compute a function $f : \mathbb{N} \to \mathbb{N}$, for some seed that encodes a number $a \in \mathbb{N}$, the tile system should produce a unique final configuration which encodes $f(a)$. To allow configurations to encode numbers, I will designate each tile as a 1- or a 0-tile and define an ordering on the positions of the tiles. The ith bit of a is 1 iff the tile in the ith position of the seed configuration is a 1-tile (note that $empty$ will almost always be a 0-tile). I will also require that the seed contains tiles coding for bits of a and no more than a constant number of extraneous tiles. The final configuration may have more than one possible ordering of the attachments. Formally, let Γ and T be sets of tiles. Let Δ be the set of all possible seed configurations $S : \mathbb{Z}^2 \to \Gamma$, and let Ξ be the set of all possible finite configurations $C : \mathbb{Z}^2 \to \Gamma \cup T$. Let $v : \Gamma \cup T \to \{0, 1\}$ code each tile as a 1- or a 0-tile and let $o_s, o_f : \mathbb{N} \to \mathbb{Z}^2$ be two injections. Let the seed encoding function $e_s : \Delta \to \mathbb{N}$ map a seed S to a number such that $e_s(S) = \sum_{i=0}^{\infty} 2^i v(S(o_s(i)))$ if and only if for no more than a constant number of (x, y) not in the image of o_s, $(x, y) \in S$. Let the answer encoding function $e_f : \Xi \to \mathbb{N}$ map a configuration F to a number such that $e_f(F) = \sum_{i=0}^{\infty} 2^i v(F(o_f(i)))$.

Let $\mathbb{S} = \langle T, g, \tau \rangle$ be a tile system. I say that $\mathbb{S}$ computes a function $f : \mathbb{N} \to \mathbb{N}$ iff for all $a \in \mathbb{N}$ there exists a seed configuration S such that $\mathbb{S}$ produces a unique final configuration F on S and $e_s(S) = a$ and $e_f(F) = f(a)$.

I generalize the above definition for a larger set of functions. Let $\hat{m}, \hat{n} \in \mathbb{N}$ and $f : \mathbb{N}^{\hat{m}} \to \mathbb{N}^{\hat{n}}$ be a function. For all $0 \leq m < \hat{m}$ and $0 \leq n < \hat{n}$, let $o_{s_m}, o_{f_n} : \mathbb{N} \to \mathbb{Z}^2$ be injections. Let the seed encoding functions $e_{s_m} : \Delta \to \mathbb{N}$ map a seed S to $\hat{m}$ numbers such that $e_{s_m}(S) = \sum_{i=0}^{\infty} 2^i v(S(o_{s_m}(i)))$ iff for no more than a constant number of (x, y) not in the union of the images of all o_{s_m}, $(x, y) \in S$. Let the answer encoding functions $e_{f_n} : \Xi \to \mathbb{N}$ each map a configuration F to a number such that $e_{f_n}(F) = \sum_{i=0}^{\infty} 2^i v(F(o_{f_n}(i)))$. Then I say that $\mathbb{S}$ computes the function f iff for all $\vec{a} \in \mathbb{N}^{\hat{m}} = \langle a_0, a_1, \ldots, a_{\hat{m}-1}\rangle$, there exists a seed configuration S such that $\mathbb{S}$ produces a unique final configuration F on S and $e_{s_m}(S) = a_m$ and $\langle e_{f_0}(F), e_{f_1}(F), \ldots, e_{f_{\hat{n}-1}}\rangle = f(\vec{a})$.

2.1. Nondeterministic computation

Until now, I have looked only at deterministic computation. In order for a system to compute a function, I required that the system produces unique final configurations. In some implementations of the tile assembly model systems, many assemblies happen in parallel. In fact, it is often almost impossible to create only a single assembly, and thus there is a parallelism that my previous definitions did not take advantage of. Here, I define the notion of nondeterministic computation in the tile assembly model.

I have defined a tile system to produce a unique final configuration on a seed if for all sequences of tile attachments, all possible final configurations are identical. If different sequences of tile attachments attach different tiles in the same position, the system is said to be **nondeterministic**. Intuitively, a system nondeterministically computes a function iff at least one of the possible sequences of tile attachments produces a final configuration which codes for the solution.

Since a nondeterministic computation may have unsuccessful sequences of attachments, it is important to distinguish the successful ones. Further, in many implementations of the tile assembly model that would simulate all the nondeterministic executions at once, it is useful to be able to identify which executions succeeded and which failed in a way that allows selecting only the successful ones. For some problems, only an exponentially small fraction of the assemblies would represent a solution, and finding such an assembly would be difficult. For example, a DNA based crystal growing system would create millions of crystals, and only a few of them may represent the correct answer, while all others represent failed computations. Finding a successful computation by sampling the crystals at random would require time exponential in the input. Thus it would be useful to attach a special identifier tile to the crystals that succeed so that the crystals may be filtered to find the solution quickly. It may also be possible to attach the special identifier tile to solid support so that the crystals representing successful computations may be extracted from the solution. I thus specify one of the tiles of a system as an **identifier** tile that only attaches to a configuration that represents a successful sequence of attachments.

Let $\hat{m}, \hat{n} \in \mathbb{N}$ and let $f : \mathbb{N}^{\hat{m}} \to \mathbb{N}^{\hat{n}}$ be a function. For all $0 \le m < \hat{m}$ and $0 \le n < \hat{n}$, let o_{s_m}, $o_{f_n} : \mathbb{N} \to \mathbb{Z}^2$ be injections. Let the seed encoding functions $e_{s_m} : \Delta \to \mathbb{N}$ map a seed S to $\hat{m}$ numbers such that $e_{s_m}(S) = \sum_{i=0}^{\infty} 2^i v(S(o_{s_m}(i)))$ iff for no more than a constant number of (x, y) not in the union of the images of all o_{s_m}, $(x, y) \in S$. Let the answer encoding functions $e_{f_n} : \Xi \to \mathbb{N}$ each map a configuration F to a number such that $e_{f_n}(F) = \sum_{i=0}^{\infty} 2^i v(F(o_{f_n}(i)))$. Let $\mathbb{S}$ be a tile system with T as its set of tiles, and let $r \in T$. Then I say that $\mathbb{S}$ **nondeterministically computes** a function f with identifier tile r iff for all $\vec{a} \in \mathbb{N}^{\hat{m}} = \langle a_0, a_1, \ldots, a_{\hat{m}-1}\rangle$ there exists a seed configuration S such that for all final configurations F that $\mathbb{S}$ produces on S, $r \in F(\mathbb{Z}^2)$ iff $\forall 0 \le m < \hat{m}$, $e_{s_m}(S) = a_m$ and $\langle e_{f_0}(F), e_{f_1}(F), \ldots, e_{f_{\hat{n}-1}}(F)\rangle = f(\vec{a})$.

If for all $\hat{m}, \hat{n} \in \mathbb{N}$, for all $0 \le m < \hat{m}$ and $0 \le n < \hat{n}$, the o_{s_m} and o_{f_n} functions are allowed to be arbitrarily complex, the definition of computation in the tile assembly model is not very interesting because the computational intelligence of the system could simply be encoded in the o_{s_m} and o_{f_n} functions. For example, suppose h is the halting characteristic function (for all $a \in \mathbb{N}$, $h(a) = 1$ if the ath Turing machine halts on input a, and 0 otherwise) and o_{s_0} is such that the input a is encoded in some straight line if $h(a) = 1$ and in some jagged line otherwise. Then it would be trivial to design a tile system to solve the halting problem. Thus the complexities of the o_{s_m} and o_{f_n} functions need to be limited.

The problem of limiting the complexities is not a new one. When designing Turing machines, the input must be encoded on the tape and the theoreticians are faced with the exact same problem: an encoding that is too powerful could render the Turing machine capable of computing uncomputable functions. The common solution is to come up with a single straightforward encoding, e.g., for all $m \in \mathbb{N}$, converting the input element of $\mathbb{N}^m$ into an element of $\mathbb{N}$ via a mapping $\mathbb{N}^m \to \mathbb{N}$ and using the intuitive direct binary encoding of that element of $\mathbb{N}$ on the Turning machine tape for all computations [35]. A similar approach is possible in the tile assembly model, requiring all systems to start with the input encoded the same way. In fact, it has been shown that such a definition conserves Turing universality of the tile systems [39]. However, the assembly time complexity of such systems may be adversely affected. In my definitions, I wish to give the system architect freedom in encoding the inputs and outputs for the sake of efficiency of computation; however, I restrict the o_{s_m} and o_{f_n} functions to be computable in linear time on a Turing machine.

Thus these functions cannot add too much complexity-reducing power to the systems (the functions themselves cannot compute anything more complex than what linear-time algorithms can) while allowing the architects the freedom to place the inputs and outputs where they wish.

I have defined what it means for a tile system to compute a function f, whether it is deterministically or nondeterministically; however, I have not mentioned what happens if f is not defined on some arguments. Intuitively, the deterministic system should not produce a unique final configuration, and no nondeterministic assembly should contain the identifier tile. The above definitions formalize this intuitive idea.

In the remainder of this paper I will examine systems that factor positive integers; that is, systems that nondeterministically compute, for all $\zeta \geq 2$, $f(\zeta) = \langle \alpha, \beta \rangle$, such that $\alpha, \beta \geq 2$ and $\alpha\beta = \zeta$. To prevent f from being multivalued, the pair $\langle \alpha, \beta \rangle$ must be unique. For example, one could restrict $\langle \alpha, \beta \rangle$ to the largest possible $alpha$, where $\alpha \geq \beta$, though any well defined restriction will work.

I refer to such systems as factoring tile systems.

3. Factoring tile systems

In this section, I will describe two factoring tile systems. To that end, I will introduce three tile systems, building up factoring functionality one step at a time. I will then combine those systems to create a system that factors at temperature four, and then discuss simplifying this factoring system to work at temperature three. All the systems use $\Theta(1)$ distinct tiles. The factoring systems, and the proofs of their correctness, are based in part on the multiplier system (one that deterministically computes $f(\alpha, \beta) = \alpha\beta$) from [13]. Intuitively, the factoring system will nondeterministically pick two numbers, multiply them, and then compare the result to the input. If the result and the input match, the assembly will include an identifier tile.

The factoring tile system will use a set of tiles T_4. I will define this set in three disjoint subsets: $T_4 = T_{\text{factors}} \cup T_\times \cup T_\checkmark$. The tiles in T_{factors} nondeterministically "guess" two factors; $T_\times$ is identical to $T_\times$ defined in [13] and multiply the two factors; and the tiles in $T_\checkmark$ ensure the computation is complete and compare the product of the factors to the input.

Whenever considering a number $\alpha \in \mathbb{N}$, I will refer to the size of α, in bits, as n_α. I will further refer to the ith bit of α as α_i; that is, for all $i \in \mathbb{N}$, $\alpha_i \in \{0, 1\}$ such that $\sum_i \alpha_i 2^i = \alpha$. The least significant bit of α is α_0. Finally, I define $\lambda_\alpha \in \mathbb{N}_{\geq 1}$ to be the smallest positive integer such that $\alpha_{\lambda_\alpha} = 1$. For example, let $\alpha = 1000101_2$, then $n_\alpha = 7$, and $\lambda_\alpha = 2$ because the smallest positive power of 2 in 1000101_2 is 2^2. Of course, the same definitions extend to β, ζ and other variables.

The tile systems I will describe will use four types of tiles to encode the input number the system is factoring. Let the set of those tiles be $\Gamma_4 = \{\gamma_L = \langle null, null, s, null \rangle, \gamma_0 = \langle null, null, 0^\star, null \rangle, \gamma_1 = \langle null, null, 1^\star, null \rangle, \gamma_s = \langle null, null, |o|, s \rangle\}$. Fig. 1 shows a graphical representation of Γ_4.

3.1. Guessing the factors

I first describe a tile system that, given a seed consisting of just a single tile will nondeterministically pick two natural binary numbers, α and β, such that $\alpha, \beta \geq 2$, and encode them using tiles. The system will use the set of tiles T_{factors}.

Fig. 2(a) shows the concepts behind the tiles in T_{factors}. The concepts include variables a and b. Each variable can take on as values the elements of the set $\{0, 1\}$. Fig. 2(b) shows the 13 actual tile types in T_{factors}. The symbols on the sides of the tiles indicate the binding domains of those sides. The value in the middle of each tile t indicates its $v(t)$ value (some tiles have no value in the middle because their v value will not be important and can be assumed to be 0).

Lemma 3.1. *Let $\Sigma_{\text{factors}} = \{|, ||, |^\star|, |o|, 0, 1, 00, 10\}$. For all $\sigma \in \{|, ||, |^\star|, |o|\}$, let $g_{\text{factors}}(\sigma, \sigma) = 4$ (for all other $\sigma \in \Sigma_{\text{factors}}$, $g(\sigma, \sigma)$ is not important for now). Let $\tau_{\text{factors}} = 4$. Let T_{factors} be as described in Fig. 2(b). Let $\mathbb{S}_{\text{factors}} = \langle T_{\text{factors}}, g_{\text{factors}}, \tau_{\text{factors}} \rangle$. Let $S_{\text{factors}} : \mathbb{Z}^2 \to \{\gamma_s\}$ be a seed configuration with a single nonempty tile, such that $S(1, 1) = \gamma_s$ and $\forall x, y \in \mathbb{Z}$, $(x, y) \neq (1, 1) \implies S_{\text{factors}}(x, y) = empty$.*

Then for all $\alpha, \beta \geq 2$, $z \geq 0$, $\mathbb{S}_{\text{factors}}$ produces a final configuration F on S_{factors} such that:

(1) $F(1, 1) = \gamma_s$
(2) $F(1, 0) = f_{btop}$

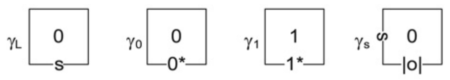


Fig. 1. There are four tiles in Γ_4. The value in the middle of each tile t represents that tile's $v(t)$ value and each tile's name is written on its left.

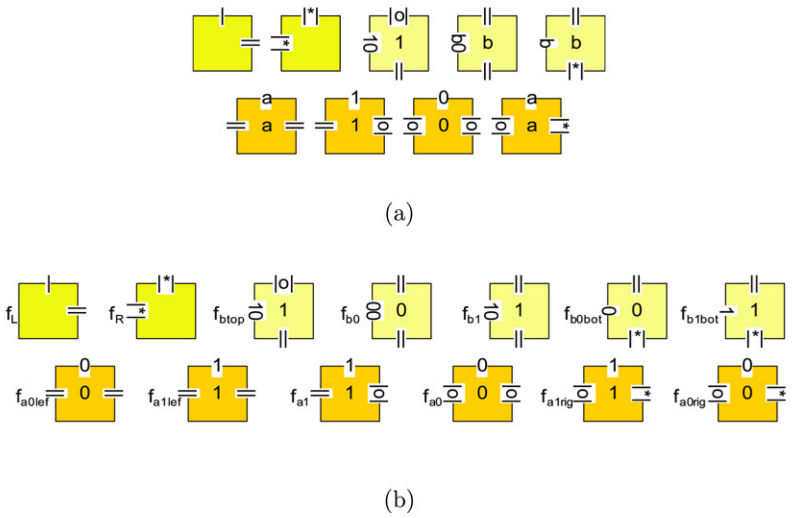


Fig. 2. The concepts behind the tiles in T_{factors} include variables a and b (a). Each variable can take on as values the elements of the set $\{0, 1\}$. There are 13 actual tile types in T_{factors} (b). The value in the middle of each tile t indicates its $v(t)$ value (some tiles have no value in the middle because their v value will not be important and can be assumed to be 0) and each tile's name is written on its left.

(3) $\forall i \in \{1, 2, \dots, n_\beta - 2\}$, $F(1, -i) = f_{b\beta_k}$, *where* $k = n_\beta - i$
(4) $F(1, -(n_\beta - 1)) = f_{b\beta_0 bot}$
(5) $F(1, -n_\beta) = f_R$
(6) $F(0, -n_\beta) = f_{a\alpha_0 rig}$
(7) $\forall i \in \{1, 2, \dots, \lambda_\alpha - 1\}$, $F(-i, -n_\beta) = f_{a0}$
(8) $F(-\lambda_\alpha, -n_\beta) = f_{a1}$
(9) $\forall i \in \{\lambda_\alpha + 1, \lambda_\alpha + 2, \dots, n_\alpha - 1\}$, $F(-i, -n_\beta) = f_{a\alpha_i lef}$
(10) $\forall i \in \{1, 2, \dots, z\}$, $F(-(n_\alpha - 1 + i), -n_\beta) = f_{a0lef}$
(11) $F(-(n_\alpha + z), -n_\beta) = f_L$

and for all final configurations F *that* $\mathbb{S}_{\text{factors}}$ *produces on* S_{factors}, F *corresponds to some choice of* $\alpha, \beta \geq 2$ *and* $z \geq 0$.

Intuitively, $\mathbb{S}_{\text{factors}}$ writes out, in order, the bits of α and β, for all $\alpha, \beta \geq 2$, and pads α with z extra 0-tiles, where $z \geq 0$. Fig. 3 shows a sample final configuration that corresponds to $\beta = 103 = 1100111_2$, $\alpha = 97 = 1100001_2$, and $z = 7$.

Proof. Let $\alpha, \beta \geq 2$, let $z \geq 0$. I first show that tiles may attach to S_{factors} to produce a final configuration F that encodes α and β and pads β with z 0-tiles.

(1) F and S_{factors} must agree at position $(1, 1)$, so $F(1, 1) = \gamma_s$.
(2) Note that $bd_S(F(1, 1)) = |o|$ and the only tile type t with $bd_N(t) = |o|$ is $f_{b\text{top}}$, so $F(1, 0) = f_{b\text{top}}$.
(3) Note that $bd_S(F(1, 0)) = ||$, and there are four types of tiles t such that $bd_N(t) = ||$ (f_{b0}, f_{b1}, $f_{b0\text{bot}}$, and $f_{b1\text{bot}}$), so only those types of tiles may attach below. Two of those tile types have $bd_S(t) = ||$ (f_{b0} and f_{b1}), so again only those four tile types may attach below them. Therefore tiles may attach such that for all $i \in \{1, 2, \dots, n_\beta - 2\}$, $F(1, -i) = f_{b\beta_k}$, where $k = n_\beta - i$.
(4) Until finally, a tile of type $f_{b0\text{bot}}$ or $f_{b1\text{bot}}$ attaches so that $F(1, -(n_\beta - 1)) = f_{b\beta_0\text{bot}}$.
(5) Note that $bd_S(F(1, -(n_\beta - 1))) = |\star|$, and the only tile type t with $bd_N(t) = |\star|$ is f_R, so $F(1, -n_\beta) = f_R$.

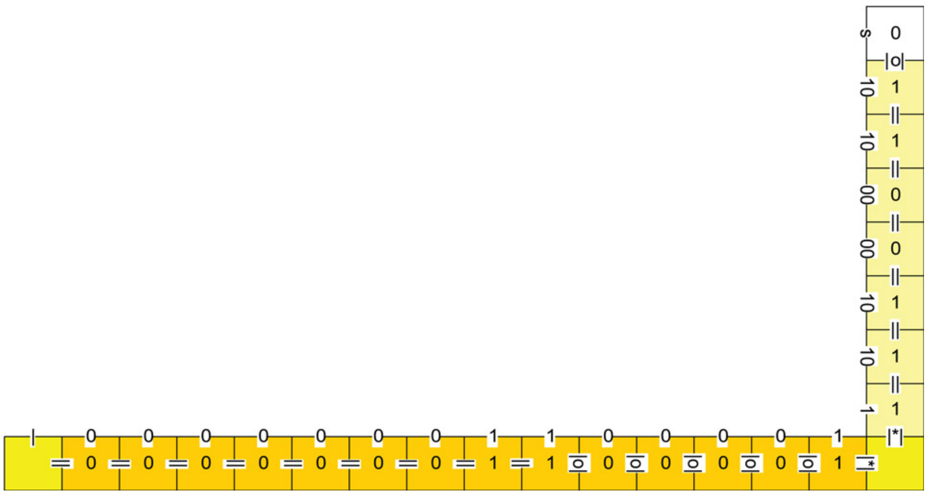


Fig. 3. $\mathbb{S}_{\text{factors}}$ produces a final configuration F on S_{factors} such that F encodes two binary numbers. In this example, F encodes $103 = 1100111_2$ and $97 = 1100001_2$. Note that the number encoded in the lower row may have leading zeros; in this case it has seven leading 0-tiles.

(6) Note that $bd_W(F(1, -n_\beta)) = |{}^\star|$, and there are two types of tiles t such that $bd_E(t) = |{}^\star|$ ($f_{a0\text{rig}}$ and $f_{a1\text{rig}}$), so only types of tiles may attach at position $(0, -n_\beta)$. Thus the correct tile may attach such that $F(0, -n_\beta) = f_{a\alpha_0\text{rig}}$.
(7) Note that $bd_W(F(0, -n_\beta)) = |o|$, and there are two types of tiles t such that $bd_E(t) = |o|$ (f_{a0} and f_{a1}), and only one of those two types has $bd_W(t) = |o|$ (f_{a0}). Therefore, for all $i \in \{1, 2, \ldots, \lambda_\alpha - 1\}$, f_{a0} can attach such that $F(-i, -n_\beta) = f_{a0}$.
(8) Until finally, a tile of type f_{a1} attaches so that $F(-\lambda_\alpha, -n_\beta) = f_{a1}$.
(9) Note that $bd_W(F(-\lambda_\alpha, -n_\beta)) = ||$, and there are three types of tiles t such that $bd_E(t) = ||$ ($f_{a0\text{lef}}$, $f_{a1\text{lef}}$, and f_L). Two of those tile types have $bd_W(t) = ||$ ($f_{a0\text{lef}}$ and $f_{a1\text{lef}}$), so again only those three tile types may attach to the left of them. Therefore, tiles may attach such that for all $i \in \{\lambda_\alpha + 1, \lambda_\alpha + 2, \ldots, n_\alpha - 1\}$, $F(-i, -n_\beta) = f_{a\alpha_i\text{lef}}$.
(10) Similarly, to the west of the position $(-(n_\alpha - 1, -n_\beta))$, $f_{a0\text{lef}}$ may attach such that for all $i \in \{1, 2, \ldots, z\}$, $F(-(n_\alpha - 1 + i), -n_\beta) = f_{a0\text{lef}}$.
(11) Until finally, a tile of type f_L attaches so that $F(-(n_\alpha + z), -n_\beta) = f_L$.

Let $v : T_{\text{factors}} \to \{0, 1\}$ be such that $v(f_{b0}) = v(f_{b0\text{bot}}) = v(f_{a0\text{lef}}) = v(f_{a0}) = v(f_{a0\text{rig}}) = v(f_L) = v(null) = 0$ and $v(f_{b\text{top}}) = v(f_{b1}) = v(f_{b1\text{bot}}) = v(f_{a1\text{lef}}) = v(f_{a1}) = v(f_{a1\text{rig}}) = 1$. For all other $t \in T_{\text{factors}}$ the $v(t)$ value does not matter and can be assumed to be either 0 or 1.

Note that:

- No more tiles may attach to F.
- $\sum_{i=0}^{\infty} v(F(1, -(n_\beta - 1 + i)))2^i = \beta$.
- $\sum_{i=0}^{\infty} v(F(-i, -n_\beta))2^i = \alpha$.
- For all $i \in \{1, 2, \ldots, z\}$, $v(F(-(n_\alpha - 1 + i), -n_\beta)) = 0$.

Thus for all choices of $\alpha, \beta \geq 2$, and $z \geq 0$, there exists a final configuration F produced by $\mathbb{S}_{\text{factors}}$ on S_{factors} that encodes α and β and pads α with z 0-tiles. Further note that for all final configurations F produced by $\mathbb{S}_{\text{factors}}$ on S_{factors}:

- F cannot encode $\beta < 2$ because $v(F(1, 0)) = 1$ implies that the most significant bit of β is 1 and $F(1, -1)$ cannot be f_R, thus β has at least two bits.
- F cannot encode $\alpha < 2$ because the tile encoding α_{λ_α} cannot be at position $(0, 1)$ and thus there exists a non-zero power of two in α.

Thus all final configurations F produced by $\mathbb{S}_{\text{factors}}$ on S_{factors} encode some $\alpha, \beta \geq 2$ and pad α with some $z \geq 0$ 0-tiles. □

For a single choice of $\alpha, \beta \geq 2$, there are several final configurations that encode α and β. They differ in the choice of z. I am interested only in two of those final configurations, for $z = n_\beta$ and for $z = n_\beta - 1$, because the product of α and β is either $n_\alpha + n_\beta$ or $n_\alpha + n_\beta - 1$ bits long.

Lemma 3.2 (*Factors Assembly Time Lemma*). *For all $\alpha, \beta \geq 2$, the assembly time of the final configuration F produced by $\mathbb{S}_{\text{factors}}$ on S_{factors} that encodes α and β and pads α with n_β or $n_\beta - 1$ 0-tiles is $\Theta(n_\alpha + n_\beta)$.*

Proof. For each tile in F to attach, a tile in a specific location must have attached before it (either to the north or to the east). Thus there is no parallelism in this assembly, and the assembly time equals the total number of tiles that attach, which is $\Theta(n_\alpha + n_\beta)$. $\square$

Lemma 3.3. *Let each tile that may attach to a configuration at a certain position attach there with a uniform probability distribution. For all $\alpha, \beta \geq 2$, let $\delta = \alpha\beta$, then the probability of assembling a particular final configuration encoding α and β and padding α with $n_\delta - n_\alpha$ 0-tiles is at least $\left(\frac{1}{4}\right)^{n_\beta} \left(\frac{1}{3}\right)^{n_\delta}$.*

Proof. I will calculate the probabilities of each tile attaching in its proper position and then multiply those probabilities together to get the overall probability of a final configuration.

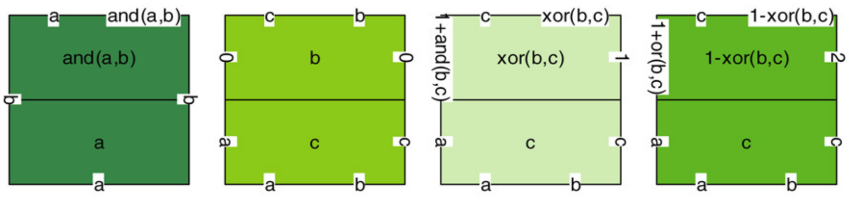
(1) The seed automatically becomes part of the final configuration with probability $p_1 = 1$.
(2) There is only one tile that may attach in position $(1, 0)$, so it attaches with probability $p_2 = 1$.
(3) For the next $n_\beta - 2$ positions, out of four possible tiles that may attach, only one is the correct one, so the probability that the next $n_\beta - 2$ tiles attach correctly is $p_3 = \left(\frac{1}{4}\right)^{n_\beta - 2}$.
(4) The tile representing the last bit of β is also one out of the possible four so the probability of it attaching is $p_4 = \frac{1}{4}$.
(5) Only one tile may attach below the 0th bit of β, so its probability of attaching is $p_5 = 1$.
(6) There are two possible tiles that may attach to represent α's 0th bit, and only one is correct, so the probability of it attaching is $p_6 = \frac{1}{2}$.
(7) The next $\lambda_\alpha - 1$ tiles must be one of two possible tiles, so the probability of all of them attaching correctly is $p_7 = \left(\frac{1}{2}\right)^{\lambda_\alpha - 1}$.
(8) The next tile encodes the smallest positive power of 2 in α and is one of two possible tiles, so its probability of attaching is $p_8 = \frac{1}{2}$.
(9) The rest of the bits of α can be encoded using one specific tile from the three possibilities, so the probability of all of them attaching correctly is $p_9 = \left(\frac{1}{3}\right)^{n_\alpha - \lambda_\alpha - 1}$.
(10) The next $n_\delta - n_\alpha$ tiles, depending on the desired final configuration, must be one of three possible tiles, so the probability of all of them attaching correctly is $p_{10} = \left(\frac{1}{3}\right)^{n_\delta - n_\alpha}$.
(11) Finally, the probability of the last tile attaching is $p_{11} = \frac{1}{3}$.

The overall probability of a specific final configuration is:

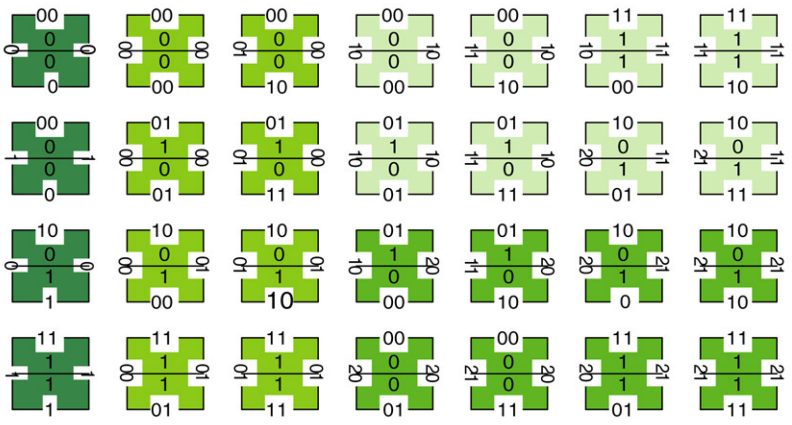
$$\prod_{i=1}^{11} p_i \geq \left(\frac{1}{4}\right)^{n_\beta - 1} \left(\frac{1}{2}\right)^{\lambda_\alpha + 1} \left(\frac{1}{3}\right)^{n_\alpha - \lambda_\alpha + n_\delta - n_\alpha} \geq \left(\frac{1}{4}\right)^{n_\beta} \left(\frac{1}{3}\right)^{n_\delta}. \quad \square$$

Corollary 3.1. *Let each tile that may attach to a configuration at a certain position attach there with a uniform probability distribution. For all $\alpha \geq \beta \geq 2$, let $\delta = \alpha\beta$, then the probability of assembling a particular final configuration encoding α and β and padding α with $n_\delta - n_\alpha$ 0-tiles is at least $\left(\frac{1}{6}\right)^{n_\delta}$.*

Proof. By Lemma 3.3, the probability of assembling a particular final configuration encoding α and β and padding α with $n_\delta - n_\alpha$ 0-tiles is at least $\left(\frac{1}{4}\right)^{n_\beta} \left(\frac{1}{3}\right)^{n_\delta}$. Because $\alpha \geq \beta$, $n_\delta \geq 2n_\beta$, so $\left(\frac{1}{4}\right)^{n_\beta} \left(\frac{1}{3}\right)^{n_\delta} \geq \left(\frac{1}{2}\right)^{n_\delta} \left(\frac{1}{3}\right)^{n_\delta} = \left(\frac{1}{6}\right)^{n_\delta}$. $\square$



(a)



(b)

Fig. 4. The concepts behind the tiles in $T_\times$ (a) include variables a, b, and c, each of which can take on as values the elements of the set $\{0, 1\}$. There are 28 actual tile types in $T_\times$ (b). Note that for each tile t, its $v(t)$ value is indicated in the upper half of the middle of the tile.

3.2. Multiplying the factors

I have just described a tile system that nondeterministically assembles the representations of two binary numbers. I will now add tiles to multiply those numbers.

Fig. 4(a) shows the concepts behind the tiles in $T_\times$. The concepts include variables a, b, and c, each of which can take on as values the elements of the set $\{0, 1\}$. Fig. 4(b) shows the 28 actual tile types in $T_\times$. These tiles are identical to the multiplier tiles from [13], thus I will reference Theorem 2.4 from [13]:

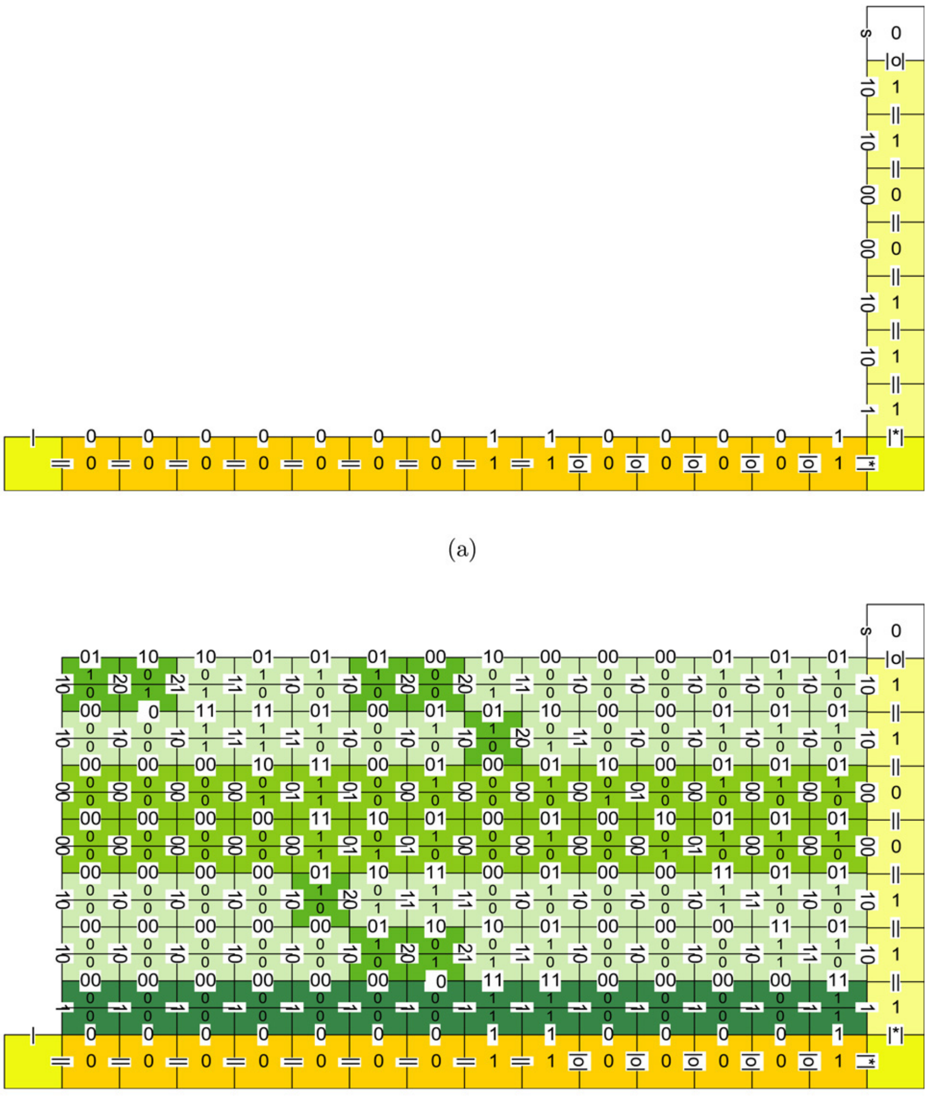
Theorem 3.1 (*Theorem 2.4 from [13]*). *Let $\Sigma_\times = \{0, 1, 00, 01, 10, 11, 20, 21\}$, for all $\sigma \in \Sigma_\times$, $g_\times(\sigma, \sigma) = 1$, $\tau_\times = 2$, and $T_\times$ be a set of tiles over $\Sigma_\times$ as described in Fig. 4(b). Then $\mathbb{S}_\times = \langle T_\times, g_\times, \tau_\times \rangle$ computes the function $f(\alpha, \beta) = \alpha\beta$.*

The reader may refer to [13] for the proof of Theorem 3.1.

Corollary 3.2. *For all $\sigma \in \Sigma_\times$, let $g'_\times(\sigma, \sigma) = 2$ and $\tau'_\times = 4$. Then $\mathbb{S}'_\times = \langle T_\times, g'_\times, \tau'_\times \rangle$ computes the function $f(\alpha, \beta) = \alpha\beta$.*

Proof. For all $\alpha, \beta \in \mathbb{N}$, $\mathbb{S}_\times$ computes the function $f(\alpha, \beta) = \alpha\beta$. Therefore, there exists some seed S_0, which encodes α and β, such that $\mathbb{S}_\times$ produces a unique final configuration F on S_0. Further, there exists at least one sequence of attachments $W = \langle \langle t_0, (x_0, y_0) \rangle, \langle t_1, (x_1, y_1) \rangle, \ldots, \langle t_k, (x_k, y_k) \rangle \rangle$ such that t_0 attaches at position (x_0, y_0) to S_0 to produce S_1, t_1 attaches at position (x_1, y_1) to S_1 to produce S_2, and so on to produce the final configuration $S_{k+1} = F$. Since for all $\sigma \in \Sigma_\times$, $g_\times(\sigma, \sigma) = 1$ and $\tau_\times = 2$, each tile t_i must have at least two non-empty neighbors to position (x_i, y_i) in S_i whose appropriate binding domains match t_i's binding domains.

By the unique final configuration corollary (Corollary 2.1 from [13]), $\mathbb{S}'_\times$ produces a unique final configuration on S_0. Since for all i, each tile t_i has at least two non-empty neighbors to position (x_i, y_i) in S_i whose appropriate binding



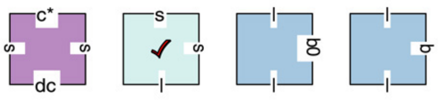
(a)

(b)

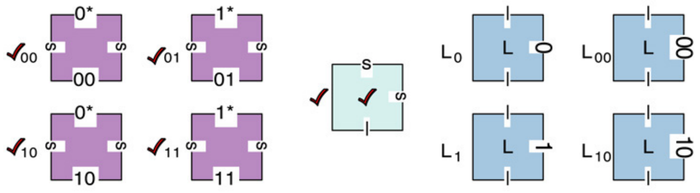
Fig. 5. A sample seed configuration $S_\times$ that encodes $\beta = 103 = 1100111_2$ and $\alpha = 97 = 1100001_2$ (a). Note that $S_\times$ is identical to a final configuration produced by $\mathbb{S}_{\text{factors}}$ on S_{factors}. $\mathbb{S}'_\times$ produces a final configuration F on $S_\times$ which encodes the product $\alpha\beta = 97 \cdot 103 = 9991 = 10011100000111_2$ along the top row (b).

domains match t_i's binding domains, and for all $\sigma \in \Sigma_\times$, $g'_\times(\sigma, \sigma) = 2$ and $\tau'_\times = 4$, the tile t_i can attach to S_i to form S_{i+1}. Thus W is a valid sequence of attachments for $\mathbb{S}'_\times$ to S_0 and $\mathbb{S}'_\times$ must produce the unique final configuration $S_{k+1} = F$ on S_0, and thus compute the function $f(\alpha, \beta) = \alpha\beta$. $\square$

Intuitively, Corollary 3.2 says that doubling the strength of every binding domain and the temperature does not alter the logic of assembly of $\mathbb{S}_\times$. To compute the product of two numbers α and β, $\mathbb{S}_\times$ attaches tiles to a seed configuration which encodes α and β, to reach a final configuration which encodes $\alpha\beta$. The logic of the system dictates that every time a tile attaches, its south and east neighbors have already attached, and its west and north neighbors have not attached [13]. Thus by doubling the strength of every binding domain and the temperature, a tile t can attach at position (x, y) in $\mathbb{S}'_\times$ iff t also attached at (x, y) in $\mathbb{S}_\times$. Thus the final configurations in the two systems will be identical, and therefore, $\mathbb{S}'_\times$ computes the function $f(\alpha, \beta) = \alpha\beta$.



(a)



(b)

Fig. 6. The concepts behind the tiles in $T_{\checkmark}$ (a) include variables b, c and d, each of which can take on as values the elements of the set $\{0, 1\}$. There are nine actual tile types in $T_{\checkmark}$; each tile's name is written on its left. The tile with a check mark in the middle will serve as the identifier tile.

Fig. 5(a) shows a sample seed configuration $S_\times$ which encodes $\beta = 103 = 1100111_2$ and $\alpha = 97 = 1100001_2$. Fig. 5(b) shows the final configuration F that $\mathbb{S}'_\times$ produces of $S_\times$. Along the top row of F, the binary number $10011100000111_2 = 9991$ encodes the product $\alpha\beta = 97 \cdot 103 = 9991$. It is not a coincidence that $S_\times$ looks exactly like a final configuration that $\mathbb{S}_{\text{factors}}$ produces on S_{factors}.

Lemma 3.4. *Let $\alpha, \beta \geq 2$, let $S_\times$ be the final configuration produced by $\mathbb{S}_{\text{factors}}$ on S_{factors} that encodes α and β and has n_β padding 0-tiles. Let F be the unique final configuration that $\mathbb{S}'_\times$ produces on $S_\times$. Let $v : T_\times \to \{0, 1\}$ be defined as in Fig. 5. Then F encodes the binary number $\delta = \alpha\beta = \sum_{i=0}^{\infty} v(F(-i, 0))2^i$. That is, the ith bit of δ is $v(F(-i, 0))$. Further, if $\alpha\beta$ has only $n_\alpha + n_\beta - 1$ bits, then it is sufficient to pad S_{factors} with $n_\beta - 1$ 0-tiles.*

The lemma follows directly from the proof of Corollary 3.2 and Theorem 3.1. $S_\times$ forms the sides of a rectangle that has the same binding domains internal to the rectangle as the seed that $\mathbb{S}_\times$ uses in [13]. $\mathbb{S}'_\times$ produces, on $S_\times$ the final configuration F, which encodes the product of the inputs on precisely the 0th row, with the ith bit of the product being $v(F(0, -i))$. Note that the product of two binary numbers α and β will always have either $n_\alpha + n_\beta$ or $n_\alpha + n_\beta - 1$ bits.

Lemma 3.5 (*Multiplication Assembly Time Lemma*). *For all $\alpha, \beta \geq 2$, the assembly time for $\mathbb{S}_\times$ on a seed encoding α and β and padding α with n_β or $n_\beta - 1$ 0-tiles is $\Theta(n_\alpha + n_\beta)$.*

This lemma follows directly from the assembly time corollary (Corollary 2.2 from [13]).

3.3. Checking the product

I have described two systems: one to produce two random numbers and another to multiply them. I now present one more system with two jobs: to make sure the multiplication is complete and to compare an input to the result of the multiplication.

Fig. 6(a) shows the concepts behind the tiles in $T_{\checkmark}$. The concepts include variables b, c and d, each of which can take on as values the elements of the set $\{0, 1\}$. Fig. 6(b) shows the nine actual tile types in $T_{\checkmark}$. The tile with a check mark in the middle will serve as the identifier tile.

Lemma 3.6. *Let $\Sigma_{\checkmark} = \{|, 0, 1, 00, 01, 10, 11, 0^\star, 1^\star, s\}$. For all $\sigma \in \{0^\star, 1^\star, s\}$, let $g_{\checkmark}(\sigma, \sigma) = 1$ and for all $\sigma' \in \{|, 0, 1, 00, 01, 10, 11\}$, let $g_{\checkmark}(\sigma', \sigma') = 2$. Let $\tau_{\checkmark} = 4$. Let $T_{\checkmark}$ be as described in Fig. 6(b). For all $\alpha, \beta, \zeta \geq 2$, let $\delta = \alpha\beta$, let $S_\times$ be the final configuration produced by $\mathbb{S}_{\text{factors}}$ on S_{factors} that encodes α and β and has $n_\delta - n_\alpha$ padding 0-tiles. Let $F_\times$ be the unique final configuration that $\mathbb{S}'_\times$ produces on $S_\times$. Let $S_{\checkmark}$ be such that:*

- $\forall x, y \in \mathbb{Z}$, $F_\times(x, y) \neq empty \implies S_\checkmark(x, y) = F_\times(x, y)$.
- $S_\checkmark(-n_\zeta, 2) = \gamma_L$.
- $\forall i \in \{0, 1, \ldots, n_\zeta - 1\}$, $S_\checkmark(-i, 2) = \gamma_{\zeta_i}$.

Let $\mathbb{S}_\checkmark = \langle T_\checkmark, g_\checkmark, \tau_\checkmark \rangle$. Then $\mathbb{S}_\checkmark$ produces a unique final configuration $F_\checkmark$ on $S_\checkmark$ and $\alpha\beta = \zeta$ iff $F_\checkmark(-n_\zeta, 1) = \checkmark$.

Proof. First, observe that if the $\checkmark$ tile is ever to attach, it must attach in position $(-n_\zeta, 1)$ because the sum of the g values of the binding domains of $\checkmark$ is exactly 4, so it must match its neighbors on all sides with non-null binding domains to attach at temperature 4, and because the only tile with a south binding domain s, matching $\checkmark$'s north binding domain, is γ_L, and it can only occur in the seed, and only in position $(-n_\zeta, 2)$.

Consider $F_\checkmark$. Working backwards, for a $\checkmark$ tile to attach at position $(-n_\zeta, 1)$, the tile directly south, at position $(-n_\zeta, 0)$ must have a north binding domain $|$. Therefore, that tile is one of the following four tiles: $\{L_0, L_1, L_{00}, L_{10}\}$. For any one of those four tiles to attach, its east neighbor's west binding domain must be 0, 1, 00 or 10, meaning two things: the first bit of the binding domain cannot be 2, which implies that the carry bit of the tile to the east is 0, and the second bit of the binding domain cannot be 1, which implies that shifted α has not run past the west bound of the computation. Together, those two properties ensure that the multiplication is proceeding correctly. In turn, the tile directly south of that position, at position $(-n_\zeta, -1)$, by the same reasoning, must be one of those four tiles and the tile to its east has the two above properties. And so on, all the tiles in the column $-n_\zeta$ must be one of those four tiles, until the position $(-n_\zeta, -n_\beta)$, where the tile must be f_L (by the definition, $S_\checkmark(-n_\delta, -n_\beta) = f_L$). Therefore, the $\checkmark$ tile may attach only if $n_\delta = n_\zeta$ and every row of the multiplication has a 0 carry bit and has not overshifted α.

Working backwards again, for a $\checkmark$ tile to attach at position $(-n_\zeta, 1)$ in $F_\checkmark$, the tile directly east must have a west binding domain s, and thus must be one of the following four tiles: $\{\checkmark_{00}, \checkmark_{01}, \checkmark_{10}, \checkmark_{11}\}$. For each of those tiles, the first digit of its north binding domain matches the second digit of its south binding domain. Therefore, $v(F_\checkmark(-(n_\zeta - 1), 0))$ must equal $v(F_\checkmark(-(n_\zeta - 1), 2))$. The same argument holds for the tile to the east of that tile, and so on until position $(1, 1)$, where $F_\checkmark(1, 1) = \gamma_s$. By Lemma 3.4, if $\delta = \alpha\beta$, then $\forall i \in \mathbb{N}$, $v(S_\checkmark(-i, 0)) = \delta_i$. Therefore, a $\checkmark$ tile may attach at position $(-n_\zeta, 1)$ to $F_\checkmark$ if and only if $\gamma = \alpha\beta$. □

Fig. 7(a) shows a sample seed $S_\checkmark$ for $\beta = 103 = 1100111_2$, $\alpha = 97 = 1100001_2$, and $\zeta = 9991 = 10011100000111_2$. Fig. 7(b) shows the final configuration $F_\checkmark$ that $\mathbb{S}_\checkmark$ produces on $S_\checkmark$. Because $\zeta = \alpha\beta$, $F_\checkmark$ contains the tile $\checkmark$.

Lemma 3.7 (*Checking Assembly Time Lemma*). *For all $\alpha, \beta, \zeta \geq 2$, if $\zeta = \alpha\beta$ then the assembly time of the final configuration F produced by $\mathbb{S}_\checkmark$ on $S_\checkmark$ that encodes α, β, and ζ and pads α with n_β or $n_\beta - 1$ 0-tiles is $\Theta(n_\zeta)$.*

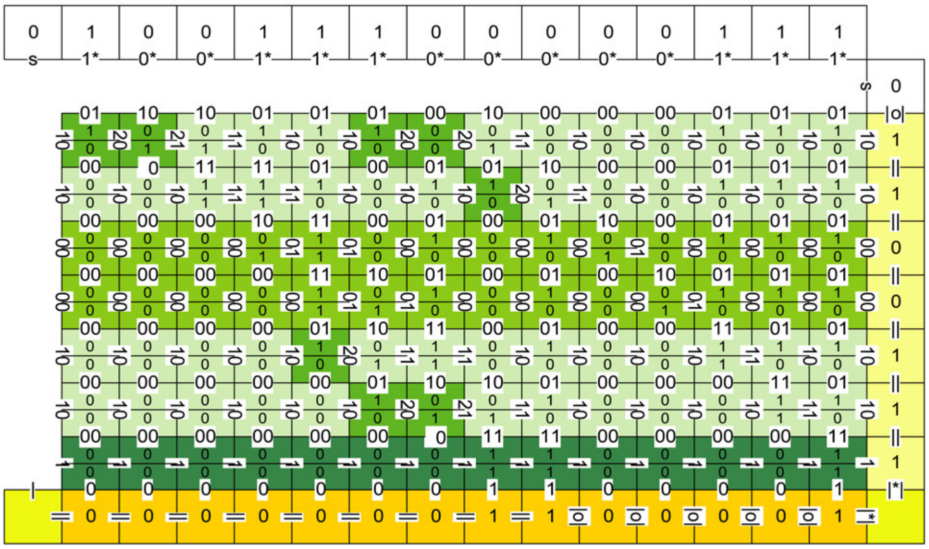
Proof. Each tile in the 0th row of F that attaches requires one other specific tile to attach first. The same is true for column n_ζ. Thus there is no parallelism in each of those two assembling processes, and the assembly time for each process equals the total number of tiles that attach in that process, thus the overall assembly time is $\Theta(\max(n_\zeta, n_\beta)) = \Theta(n_\zeta)$. □

3.4. Factoring at temperature four

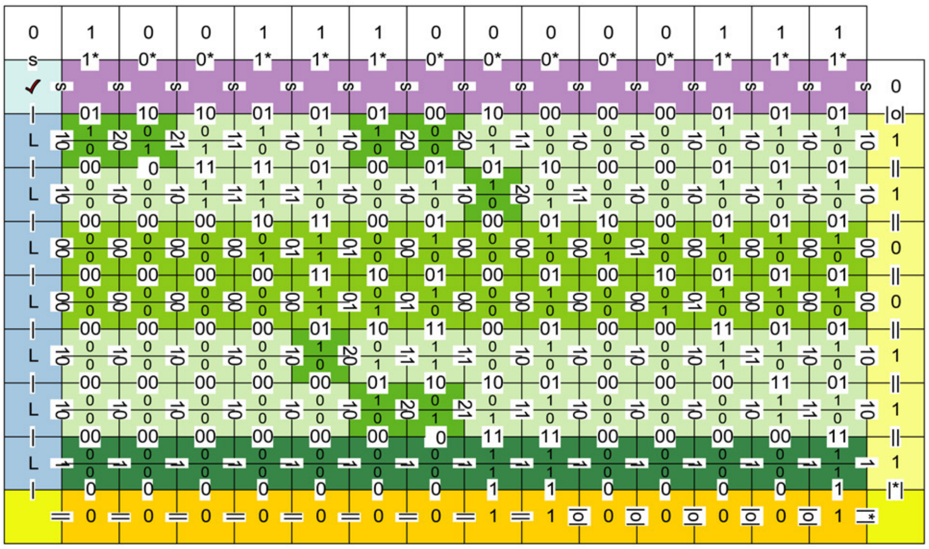
I have defined three different systems that perform the necessary pieces of factoring a number. I now put them together into a single system $\mathbb{S}_4$ and argue that $\mathbb{S}_4$ is a factoring system.

Theorem 3.2 (*Temperature Four Factoring Theorem*). *Let $\Sigma_4 = \Sigma_{\text{factors}} \cup \Sigma_\times \cup \Sigma_\checkmark$. Let $T_4 = T_{\text{factors}} \cup T_\times \cup T_\checkmark$. Let g_4 be such that g_4 agrees with g_{factors}, $g'_\times$, and $g_\checkmark$ on their respective domains (note that for all elements of the domains of more than one of those functions, those functions agree). Let $\tau_4 = 4$. Then the system $\mathbb{S}_4 = \langle T_4, g_4, \tau_4 \rangle$ is a factoring tile system.*

If a tile system is the combination of three distinct tile systems, the behavior of the large system need not be the combined behavior of the three smaller systems — the tiles from different systems can interfere with each other. However, I have designed $\mathbb{S}_{\text{factors}}$, $\mathbb{S}'_\times$, and $\mathbb{S}_\checkmark$ to work together, without interfering. For the most part, each system uses a disjoint set of binding domains, sharing binding domains only where tiles from the different systems are designed to interact. As a result, tiles from each system have a particular set of positions where they can attach: tiles from T_{factors} can only attach in column 1 and row $-n_\beta$, tiles from $T_\times$ can only attach in the rectangle defined by the



(a)



(b)

Fig. 7. A sample seed configuration $S_{\checkmark}$ that encodes $\beta = 103 = 1100111_2$, $\alpha = 97 = 1100001_2$, and $\zeta = 9991 = 10011100000111_2$ (a). $\mathbb{S}_{\checkmark}$ produces a final configuration F on $S_{\checkmark}$, which includes a $\checkmark$ tile iff $\zeta = \alpha\beta$ (b).

rows 0 and $-(n_\beta - 1)$ and columns 0 and $-(n_\zeta - 1)$, and tiles from $T_{\checkmark}$ can only attach in column $-n_\zeta$ and row 1, thus the tiles do not interfere with each other.

Proof of Theorem 3.2. For all $\zeta \geq 2$, Let S_4 be such that

- $S_4(1, 1) = \gamma_s$.
- $S_4(-n_\zeta, 2) = \gamma_L$.

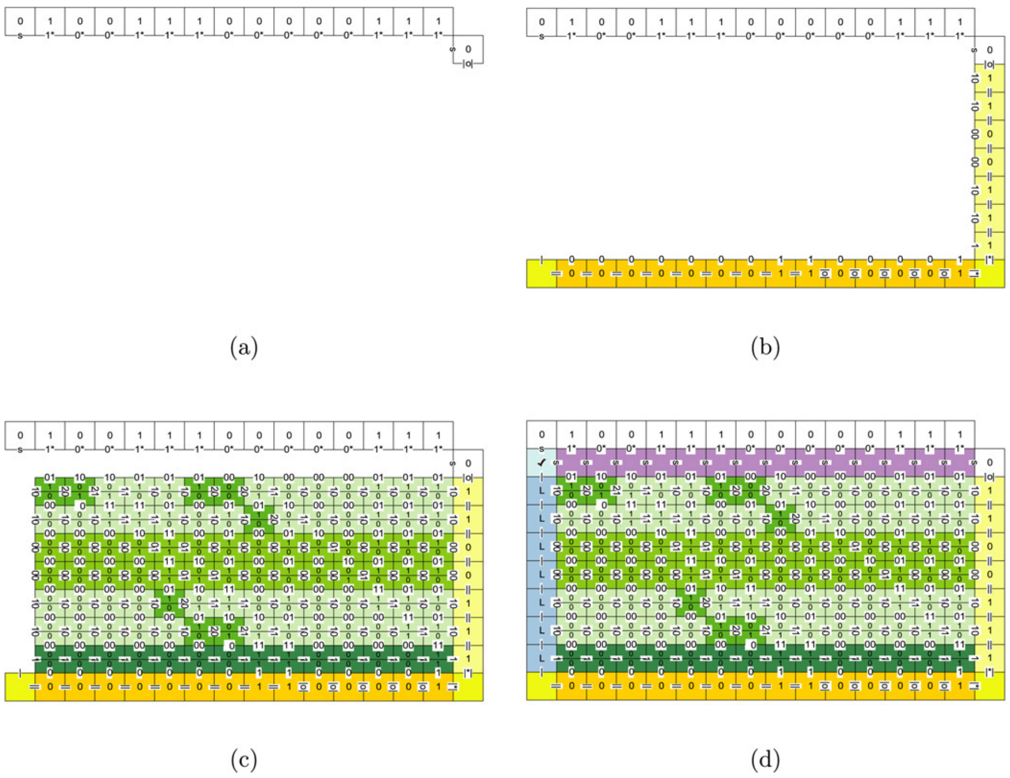


Fig. 8. $\mathbb{S}_4$ factors a number ζ encoded in the seed configuration, e.g. $\zeta = 9991 = 10011100000111_2$ (a). In one sequence of possible attachments, encodings of two randomly selected numbers α and β attach nondeterministically to the seed such that $\alpha, \beta \geq 2$, e.g. $\alpha = 97 = 1100001_2$, $\beta = 103 = 1100111_2$ (b). $\mathbb{S}_4$ then deterministically multiplies α and β to encode $\delta = \alpha\beta$ (c), and then ensures the multiplication is complete and compares δ to ζ to check if they are equal. If they are equal, a special ✓ tile attaches to signify that the factors have been found (d).

- $\forall i \in \mathbb{N}$ such that $0 \leq i < n_\zeta$, $S_4(-i, 2) = \gamma_{\zeta_i}$

For all binary numbers $\alpha, \beta \geq 2$, and for all $z \geq 0$, tiles from T_4 will attach to S_4 as follows: the tiles from T_{factors} nondeterministically attach to encode two binary numbers: α in column 1 and β in row $-n_\beta$, padding α with z 0-tiles, as described in Lemma 3.1. By Lemma 3.4 the tiles from $T_\times$ attach, in the rectangle defined by the rows 0 and $-(n_\beta - 1)$ and columns 0 and $-(n_\alpha + z - 1)$, to encode $\delta = \alpha\beta$ in the top row. Finally, tiles from $T_{\checkmark}$ attach, in column $-n_\zeta$ and row 1, such that, by Lemma 3.6 the ✓ tile only attaches if $\delta = \zeta$ and $z = n_\zeta - n_\alpha \in \{n_\beta, n_\beta - 1\}$. Let the identifier tile be ✓. Thus only if there exists a choice of $\alpha, \beta \geq 2$ such that $\alpha\beta = \zeta$ will the identifier tile attach. Further, if the identifier tile does attach to a configuration F_4, then F_4 encodes α and β as defined in Lemma 3.1. Therefore, $\mathbb{S}_4$ nondeterministically and identifiably computes the function $f(\zeta) = \langle\alpha, \beta\rangle$. □

Fig. 8(a) shows the seed configuration S_4 for $\zeta = 9991 = 1100000111_2$. Fig. 8(b)–(d) show one possible progression of tiles attaching to produce a final configuration that $\mathbb{S}_4$ produces on S_4.

While it is possible for tiles to attach in exactly the order shown in Fig. 8, other orders of attachments are also possible. For example, some tiles in $T_\times$ may attach before some other tiles in T_{factors} do, but they do not interfere with each other because they attach in disjoint sets of positions. Fig. 9 shows an alternative possible progression of tiles attaching to the seed configuration to produce a final configuration that $\mathbb{S}_4$ produces on S_4.

I have shown the details of the final configuration that finds factors α and β of ζ. It is also interesting to observe what happens when the configuration encodes an α and β whose product does not equal ζ or when α is padded with the wrong number of 0-tiles. Fig. 10(a) demonstrates an attempted assembly on choices of $\alpha = 91 = 1011011_2$ and $\beta = 84 = 1010100_2$. The multiplication finds the product $\alpha\beta = 7644 = 1110111011100_2$ and because $7644 \neq \zeta$, the tiles along row 1 do not attach and thus the ✓ tile cannot attach. Fig. 10(b) encodes the correct choices of α and β;

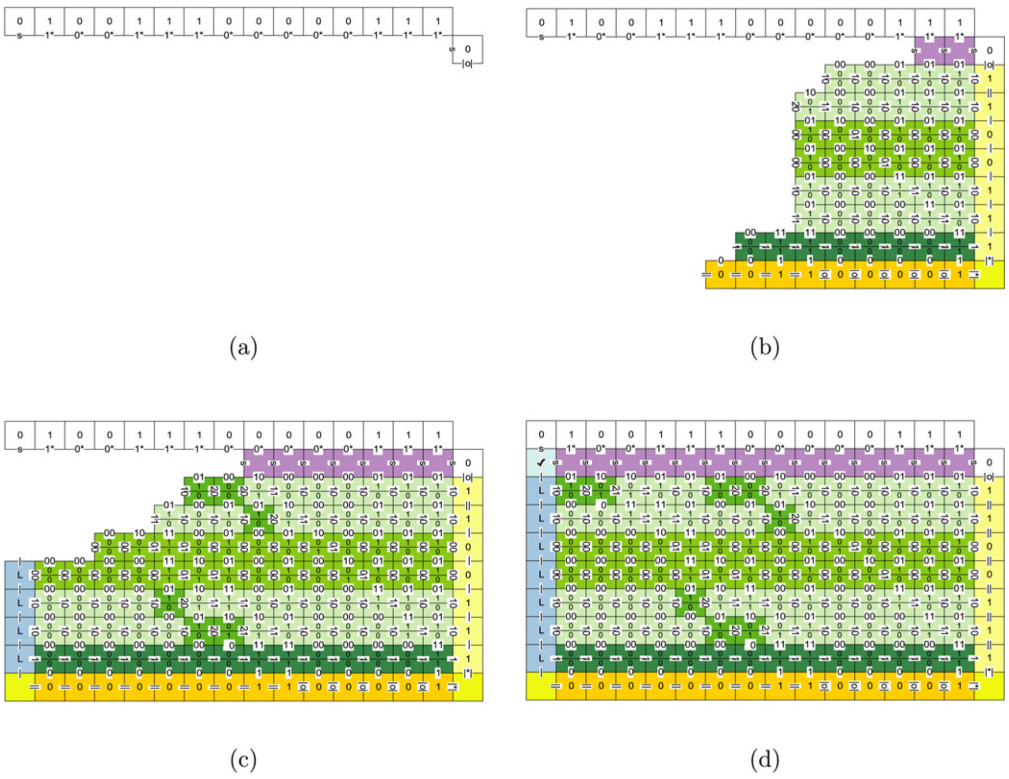


Fig. 9. $\mathbb{S}_4$ factors a number ζ encoded in the seed configuration, e.g. $\zeta = 9991 = 10011100000111_2$ (a). In a sequence of possible attachments alternate to that in Fig. 8, some tiles from all three sets T_{factors}, $T_\times$, and $T_\checkmark$ attach without waiting for all the attachments from the others sets to complete (b) and (c). Each set's tiles have designated areas of attachment and cannot attach outside of those areas, so they do not interfere with each other. Finally, for each choice of α, $\beta \geq 2$ and $z \geq 0$, all sequences of attachments result in the unique final configuration for those α, β, and z, containing a special $\checkmark$ tile iff $\alpha\beta = \zeta$ and $z = n_\zeta - n_\alpha$ (d).

however, it does not pad the α with enough 0-tiles. The multiplication cannot complete, and the tiles along the west column do not attach and thus the $\checkmark$ tile cannot attach.

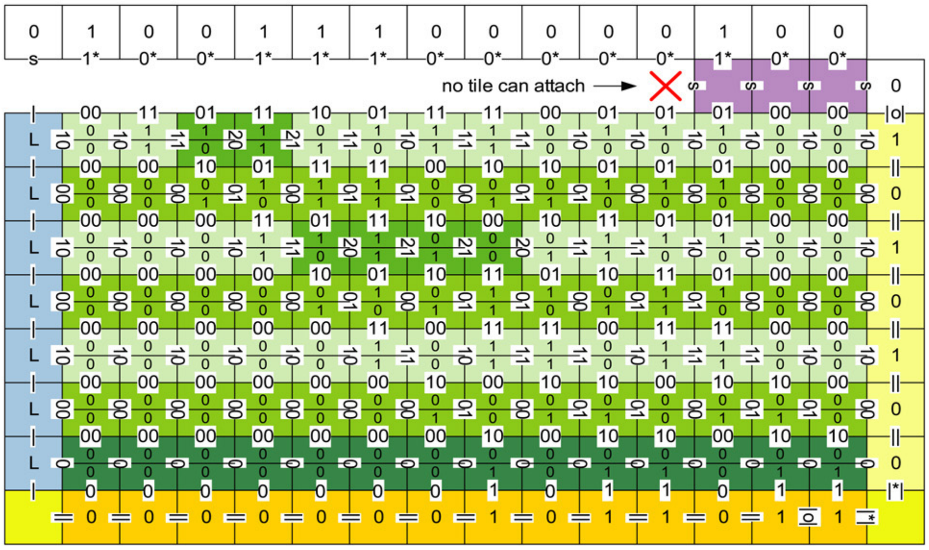
I now examine the assembly time of $\mathbb{S}_4$ and the fraction of nondeterministic assemblies that will produce the factors (or the probability of finding the factors).

Lemma 3.8 (*Factoring Assembly Time Lemma*). *For all $\alpha, \beta, \zeta \geq 2$ such that $\alpha\beta = \zeta$, the assembly time for $\mathbb{S}_4$ to produce a final configuration F that encodes α and β is $\Theta(n_\zeta)$.*

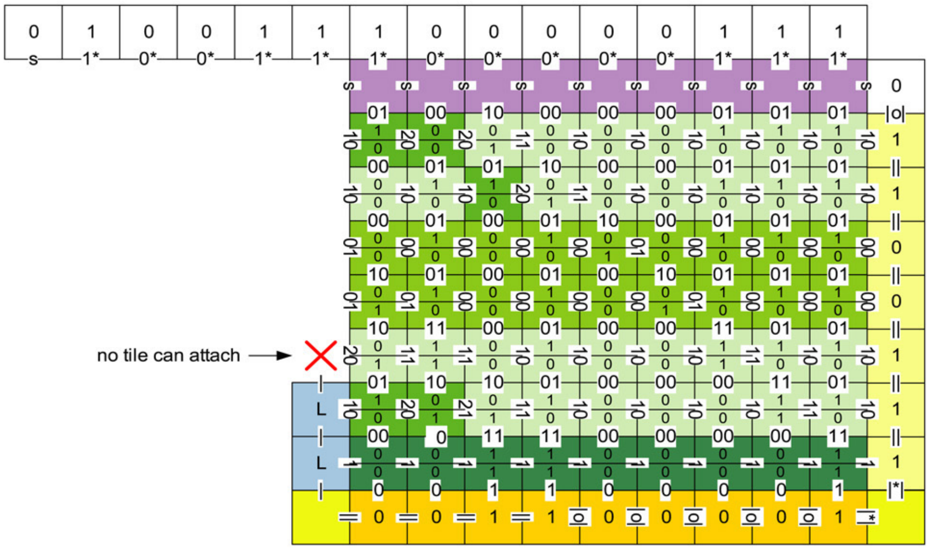
Proof. By the factors assembly time lemma (Lemma 3.2), the encoding of α and β will take $\Theta(n_\alpha + n_\beta) = \Theta(n_\zeta)$ steps. By the multiplication assembly time lemma (Lemma 3.5), multiplying α and β will take $\Theta(n_\alpha + n_\beta) = \Theta(n_\zeta)$ steps. By the checking assembly time lemma (Lemma 3.7), checking if $\alpha\beta = \zeta$ will take $\Theta(n_\zeta)$ steps. When working together, these systems do not affect each other's speed (though they may work in parallel), so the assembly time for $\mathbb{S}_4$ to produce a final configuration F that encodes α and β is $\Theta(n_\zeta)$. □

Lemma 3.9 (*Probability of Assembly Lemma*). *For all $\zeta \geq 2$, given the seed S_4 encoding ζ, assuming each tile that may attach to a configuration at a certain position attaches there with a uniform probability distribution, the probability that a single nondeterministic execution of $\mathbb{S}_4$ finds $\alpha, \beta \geq 2$ such that $\alpha\beta = \zeta$ is at least $\left(\frac{1}{6}\right)^{n_\zeta}$.*

Proof. In the worst case, a composite number ζ has only two prime factors (counting multiplicity). Either of the two factors could be α and the other β, but assume $\alpha \geq \beta$ (by forcing the larger factor to be α, I underestimate the probability). By Corollary 3.1, the probability p of assembling a particular configuration encoding α and β and padding α with $n_\zeta - n_\alpha$ 0-tiles is at least $\left(\frac{1}{6}\right)^{n_\zeta}$, as long as $\zeta = \alpha\beta$. □



(a)



(b)

Fig. 10. If $\alpha\beta \neq \zeta$ or the number of 0-tiles padding α is incorrect, the ✓ tile cannot attach. In (a), the product of $91 = 1011011_2$ and $84 = 1010100_2$ does not equal $9988 = 10011100000100_2$, so the tiles along the 0th row do not attach. In (b), α is padded with too few 0-tiles and the tiles in the west column do not attach. Note that both these configurations are final configurations.

The implication of the probability of assembly lemma (Lemma 3.9) is that a parallel implementation of $\mathbb{S}_4$, such as a DNA implementation like those in [8,33], with 6^{n_ζ} seeds has at least a $1 - \frac{1}{e} \geq 0.5$ chance of finding ζ's factors and one with $100(6^{n_\zeta})$ seeds has at least a $1 - \left(\frac{1}{e}\right)^{100}$ chance.

The lemma provides a lower bound on the probability of a successful assembly achievable by a distribution of

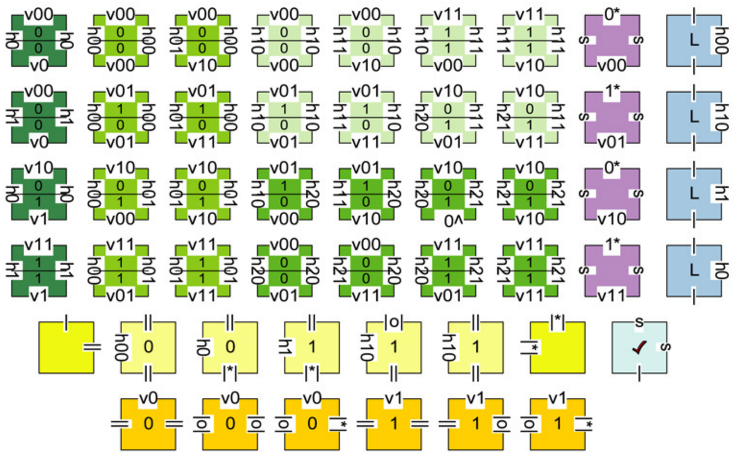


Fig. 11. In order to factor at temperature three, some of the tiles from T_4 had to be modified with new binding domains. The new tiles form the set T_3.

tile attachment probabilities. I can derive an upper bound on that probability, for large ζ, by arguing that as α and β become large, the dominant majority of the tiles in T_{factors} that must attach to encode α are f_{a0lef} and f_{a1lef} and to encode β are f_b0 and f_b1. Because the distribution of 0 and 1 bits in a random number is even, and at least one of α and β must become large as ζ becomes large, for all probability distributions of tile attachments, the probability that a single nondeterministic execution of $\mathbb{S}_4$ finds $\alpha, \beta \geq 2$ such that $\alpha\beta = \zeta$ is at most $\left(\frac{1}{2}\right)^{n_\zeta}$.

Note that T_4 contains 50 distinct tiles.

3.5. *Factoring at temperature three*

I have described how $\mathbb{S}_4$ factors at temperature four. It was simpler to explain how $\mathbb{S}_4$ works at temperature four, mostly because I could double the strength of every binding domain and the temperature of $\mathbb{S}_\times$ from [13]; however, it is actually possible to use roughly the same tiles to compute at temperature three, by altering the glue strength function. The key observation is that no tile needs all four of its neighbors in a configuration in order to attach. The most a tile needs is three, e.g. the ✓ tile.

I will need to differentiate some of the east–west binding domains from their previously identical north–south binding domain counterparts. That is, suppose that some of the binding domains of the tiles in T_4 were prefixed by v (for vertical) or h (for horizontal). Formally, let $\Sigma_3 = \{|o|, ||, |^\star|, |, s, v0, v1, v00, v01, v10, v11, 0^\star, 1^\star, h0, h1, h00, h01, h10, h11, h20, h21\}$. The strengths of the binding domains need to be defined such that each tile attaches only under the same conditions under which it attached in $\mathbb{S}_4$. The tiles in T_{factors} must attach with only one neighbor present, so for all $\sigma \in \{|o|, ||, |^\star|\}$, let $g_3(\sigma, \sigma) = 3$. The tiles in $T_\times$ must attach when two of their neighbors, the south and the east neighbors, are present so for all $\sigma \in \{h0, h1, h00, h01, h10, h11, h20, h21\}$, let $g_3(\sigma, \sigma) = 2$ and for all $\sigma' \in \{v0, v1, v00, v01, v10, v11\}$, let $g_3(\sigma', \sigma') = 1$. (Note that the alternative – making the horizontal binding domains have strength 1 and vertical strength 2 – would work for the tiles in $T_\times$, but would not work for the tiles in $T_\checkmark$.) The tiles L_0, L_1, L_{00}, and L_{10} must attach when two of their neighbors are present, and since their east binding domains have already been defined to be of strength 2, let $g_3(|, |) = 1$. Finally, the other tiles in $T_\checkmark$ only attach when three of their neighbors are present, and all their binding domains so far have been defined to be of strength 1, so for all $\sigma \in \{0^\star, 1^\star, s\}$, let $g_3(\sigma, \sigma) = 1$. Fig. 11 shows the tiles with the new binding domains labeled.

Theorem 3.3 (*Temperature Three Factoring Theorem*). *Let T_3 be as defined in Fig. 11. Let $\tau_3 = 3$. Then $\langle T_3, g_3, \tau_3\rangle$ is a factoring tile system.*

Proof. Let $\alpha, \beta, \zeta \geq 2$, and let $S3_0$ be the seed that encodes ζ as in Theorem 3.2. Then $\zeta = \alpha\beta$ iff $\mathbb{S}_4$ produces some final configuration F on $S3_0$ that encodes α and β and contains the identifier tile ✓ and all final configurations that contain ✓ encode factors of ζ.

Let W be a sequence of attachments in $\mathbb{S}_4$ that produces F on $S3_0$. $W = \langle\langle t_0, (x_0, y_0)\rangle, \langle t_1, (x_1, y_1)\rangle, \ldots, \langle t_k, (x_k, y_k)\rangle\rangle$ such that t_0 attaches at position (x_0, y_0) to $S3_0$ to produce $S3_1$, t_1 attaches at position (x_1, y_1) to $S3_1$ to produce $S3_2$, and so on to produce the final configuration $S3_{k+1} = F$.

Remember that g_3 is designed such that every tile in T_3 attaches in $\mathbb{S}_3$, under and only under the same conditions as its counterpart tile in T_4 attached in $\mathbb{S}_4$. Thus the sequence of attachments W is also a valid sequence of attachments for $\mathbb{S}_3$. Therefore, on seed $S3_0$, encoding ζ, for all final configurations F produced by $\mathbb{S}_3$ on $S3_0$, F contains ✓ iff F encodes α and β such that $\zeta = \alpha\beta$. $\square$

Lemma 3.10. *For all $\alpha, \beta, \zeta \geq 2$ such that $\alpha\beta = \zeta$, the assembly time for $\mathbb{S}_4$ to produce a final configuration F that encodes α and β is $\Theta(n_\zeta)$.*

Lemma 3.11. *For all $\zeta \geq 2$, given the seed S encoding ζ, assuming uniform distributions of all tiles, the probability that a single nondeterministic execution of $\mathbb{S}_3$ finds $\alpha, \beta \geq 2$ such that $\alpha\beta = \zeta$ is at least $\left(\frac{1}{6}\right)^{n_\zeta}$.*

Lemmas 3.10 and 3.11 follow directly from the factoring assembly time lemma (Lemma 3.8) and the probability of assembly lemma (Lemma 3.9), respectively, and the fact that $\mathbb{S}_3$ follows the exact same logic as $\mathbb{S}_4$.

As with $\mathbb{S}_4$, $\mathbb{S}_3$ uses 50 tile types.

4. Contributions

The tile assembly model is a formal model of self-assembly and crystal growth. In [13], I explored what it means for a system to compute a function via deterministic assembly and identified two important quantities: the number of tile types and the assembly speed of the computation. Here, I have extended the definition of a tile system computing a function to include nondeterministic computation, adopted the two measures to these new computations, and identified a third important measure: the probability of a nondeterministic assembly identifying the solution. I explored these measures for systems that factor numbers, that is, nondeterministically compute the function $f(\zeta) = \langle\alpha, \beta\rangle$, such that $\alpha, \beta, \zeta \geq 2$ and $\alpha\beta = \zeta$. I defined factoring tile systems at temperatures four and three that use $\Theta(1)$ input tile types, $\Theta(1)$ computing tile types, and assemble in $\Theta(n_\zeta)$ steps with probability of each assembly finding the solution at least $\left(\frac{1}{6}\right)^{n_\zeta}$. Thus a parallel implementation of $\mathbb{S}_4$, such as a DNA implementation like those in [8,33], with 6^{n_ζ} seeds has at least a $1 - \frac{1}{\mathrm{e}}$ chance of finding ζ's factors, and one with $100(6^{n_\zeta})$ seeds has at least a $1 - \left(\frac{1}{\mathrm{e}}\right)^{100}$ chance. Experiments in DNA self-assembly commonly contain on the order of $10^{17} \approx 6^{22}$ assemblies [17,20,28,24]. However, those experiments in no way required a high concentration of assemblies and no specific attempts to achieve a maximum concentration were made. In fact, experiments such as these commonly try to limit the number of parallel assemblies, as all the assemblies are identical and creating many of them is simply a waste of material. Thus it is likely that orders of magnitude larger volumes of solutions orders of magnitude more concentrated than those can be achieved.

The theoretical analysis of systems that can assemble shapes or compute simpler functions [34,4,5,23,13] has led to experimental manifestations of such systems using DNA tiles [8,33]. The field of DNA computation demonstrated that DNA can be used to compute [1], solving NP-complete problems such as the satisfiability problem [12,11]. Most DNA computation suffers from high error-rates; while DNA can solve a 20 variable 3-SAT problem, it seems unlikely that the same approach would solve a much larger problem [11] because of the additive nature of error rates. Self-assembly has yielded results in error-correction [42,10,21,27,41] that may be used to decrease the error rates. Most experimental results in DNA computation have not appealed to the advantages of crystal growth; however, these early works on the fundamentals of self-assembly and the physical experimental evidence of actual DNA crystals suggest a bright future for DNA computation.

Outside of molecular computation, early investigations into programming large distributed computer networks by representing each computer as a tile in a tile assembly model system have revealed promising possibilities. For example, an implementation of a tile assembly model system on an Internet-sized network can control error-rates while inheriting three properties of self-assembly systems: fault tolerance in the presence of malicious computers, scalability due to the local communication of tiles that is not achieved in other large distributed network systems, and privacy due to the fact that no tile knows a large portion of the input or the problem being solved. Early investigations into programming large distributed computer networks using the tile assembly model have revealed promising possibilities, thus, generalizing the tile assembly model as a potentially powerful tool in software architecture research [15,18,19,16].

Acknowledgements

I wish to immensely thank Dustin Reishus and Leonard Adleman for their help formalizing and stabilizing the ideas presented in this paper, for helpful discussions, and for all their other support. I also wish to thank the anonymous reviewers for their helpful comments.

References

[1] Leonard Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (1994) 1021–1024.

[2] Leonard Adleman, Towards a mathematical theory of self-assembly, Technical Report 00-722, Department of Computer Science, University of Southern California, Los Angleles, CA, 2000.

[3] Leonard Adleman, Qi Cheng, Ahish Goel, Ming-Deh Huang, Hal Wasserman, Linear self-assemblies: Equilibria, entropy, and convergence rates, in: Proceedings of the 6th International Conference on Difference Equations and Applications, ICDEA 2001, Augsburg, Germany, June 2001.

[4] Leonard Adleman, Qi Cheng, Ashish Goel, Ming-Deh Huang, David Kempe, Pablo Moisset de Espanes, Paul Rothermund, Combinatorial optimization problems in self-assembly, in: ACM Symposium on Theory of Computing, STOC02, Montreal, Quebec, Canada, 2002, pp. 23–32.

[5] Leonard Adleman, Ashish Goel, Ming-Deh Huang, Pablo Moisset de Espanes, Running time and program size for selfassembled squares, in: ACM Symposium on Theory of Computing, STOC02, Montreal, Quebec, Canada, 2001, pp. 740–748.

[6] Leonard Adleman, Jarkko Kari, Lila Kari, Dustin Reishus, On the decidability of self-assembly of infinite ribbons, in: The 43rd Annual IEEE Symposium on Foundations of Computer Science, FOCS'02, Ottawa, Ontario, Canada, November 2002, pp. 530–537.

[7] Gagan Aggarwal, Qi Cheng, Michael H. Goldwasser, Ming-Yang Kao, Pablo Moisset de Espanes, Robert T. Schweller, Complexities for generalized models of self-assembly, SIAM Journal on Computing 34 (6) (2005) 1493–1515.

[8] Robert Barish, Paul Rothemund, Erik Winfree, Two computational primitives for algorithmic self-assembly: Copying and counting, Nano Letters 5 (12) (2005) 2586–2592.

[9] Yuliy Baryshnikov, Ed G. Coffman, Petar Momcilovic, DNA-Based Computation Times, in: Springer Lecture Notes in Computer Science, vol. 3384, 2005, pp. 14–23.

[10] Yuliy Baryshnikov, Ed G. Coffman, Nadrian Seeman, Teddy Yimwadsana, Self correcting self assembly: Growth models and the hammersley process, in: Proceedings of the 11th International Meeting on DNA Computing, DNA 2005, London, Ontario, June 2005.

[11] Ravinderjit Braich, Nickolas Chelyapov, Cliff Johnson, Paul Rothemund, Leonard Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer, Science 296 (5567) (2002) 499–502.

[12] Ravinderjit Braich, Cliff Johnson, Paul Rothemund, Darryl Hwang, Nickolas Chelyapov, Leonard Adleman, Solution of a satisfiability problem on a gel-based DNA computer, in: DNA Computing: 6th International Workshop on DNA-Based Computers, DNA2000, Leiden, The Netherlands, June 2000, pp. 27–38.

[13] Yuriy Brun, Arithmetic computation in the tile assembly model: Addition and multiplication, Theoretical Computer Science 378 (2006) 17–31.

[14] Yuriy Brun, Adding and multiplying in the tile assembly model, in: Proceedings of the 4th Foundations of Nanoscience: Self-Assembled Architectures and Devices, FNANO07, Snowbird, UT, USA, April 2007.

[15] Yuriy Brun, A discreet, fault-tolerant, and scalable software architectural style for internet-sized networks, in: Proceedings of the Doctoral Symposium at the 29th International Conference on Software Engineering, ICSE07, Minneapolis, MN, USA, May 2007, pp. 83–84.

[16] Yuriy Brun, Discreetly distributing computation via self-assembly, Technical Report USC-CSSE-2007-714, Center for Software Engineering, University of Southern California, 2007.

[17] Yuriy Brun, Manoj Gopalkrishnan, Dustin Reishus, Bilal Shaw, Nickolas Chelyapov, Leonard Adleman, Building blocks for DNA self-assembly, in: Proceedings of the 1st Foundations of Nanoscience: Self-Assembled Architectures and Devices, FNANO'04, Snowbird, UT, April 2004.

[18] Yuriy Brun, Nenad Medvidovic, An architectural style for solving computationally intensive problems on large networks, in: Proceedings of Software Engineering for Adaptive and Self-Managing Systems, SEAMS07, Minneapolis, MN, USA, May 2007.

[19] Yuriy Brun, Nenad Medvidovic, Fault and adversary tolerance as an emergent property of distributed systems' software architectures, in: Proceedings of the 2nd International Workshop on Engineering Fault Tolerant Systems, EFTS07, Dubrovnik, Croatia, September 2007.

[20] Nickolas Chelyapov, Yuriy Brun, Manoj Gopalkrishnan, Dustin Reishus, Bilal Shaw, Leonard Adleman, DNA triangles and self-assembled hexagonal tilings, Journal of American Chemical Society (JACS) 126 (43) (2004) 13924–13925.

[21] Ho-Lin Chen, Ashish Goel, Error free self-assembly with error prone tiles, in: Proceedings of the 10th International Meeting on DNA Based Computers, DNA 2004, Milan, Italy, June 2004.

[22] Matthew Cook, Paul Rothemund, Erik Winfree, Self-assembled circuit patterns, in: Proceedings of the 9th International Meeting on DNA Based Computers, DNA 2004, Madison, WI, June 2003, pp. 91–107.

[23] Pablo Moisset de Espanes, Computerized exhaustive search for optimal self-assembly counters, in: The 2nd Annual Foundations of Nanoscience Conference, FNANO'05, Snowbird, UT, April 2005, pp. 24–25.

[24] Tsu Ju Fu, Nadrian C. Seeman, DNA double-crossover molecules, Biochemistry 32 (13) (1993) 3211–3220.
[25] Ming-Yang Kao, Robert Schweller, Reducing tile complexity for self-assembly through temperature programming, in: Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, FL, January 2006, pp. 571–580.
[26] Michail G. Lagoudakis, Thomas H. LaBean, 2D DNA self-assembly for satisfiability, in: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 54, 1999, pp. 141–154.
[27] John H. Reif, Sadheer Sahu, Peng Yin, Compact error-resilient computational DNA tiling assemblies, in: Proceedings of the 10th International Meeting on DNA Based Computers, DNA 2004, Milan, Italy, June 2004.
[28] Dustin Reishus, Bilal Shaw, Yuriy Brun, Nickolas Chelyapov, Leonard Adleman, Self-assembly of DNA double-double crossover complexes into high-density, doubly connected, planar structures, Journal of American Chemical Society (JACS) 127 (50) (2005) 17590–17591.
[29] Raphael M. Robinson, Undecidability and nonperiodicity for tilings of the plane, Inventiones Mathematicae 12 (3) (1971) 177–209.
[30] Paul Rothemund, Design of DNA origami, in: Proceedings of the International Conference on Computer-Aided Design, ICCAD 2005, San Jose, CA, November 2005.
[31] Paul Rothemund, Folding DNA to create nanoscale shapes and patterns, Nature 440 (2006) 297–302.
[32] Paul Rothemund, Scaffolded DNA origami: From generalized multicrossovers to polygonal networks, Nanotechnology: Science and Computation (2006) 3–21.
[33] Paul Rothemund, Nick Papadakis, Erik Winfree, Algorithmic self-assembly of DNA Sierpinski triangles, PLoS Biology 2 (12) (2004) e424.
[34] Paul W.K. Rothemund, Erik Winfree, The program-size complexity of self-assembled squares, in: Proceedings of the ACM Symposium on Theory of Computing, STOC00, Portland, OR, USA, May 2000, pp. 459–468.
[35] Michael Sipser, Introduction to the Theory of Computation, PWS Publishing Company, 1997.
[36] David Soloveichik, Erik Winfree, Complexity of self-assembled shapes, in: Proceedings of the 10th International Meeting on DNA Based Computers, DNA 2004, Milan, Italy, June 2004.
[37] Hao Wang, Proving theorems by pattern recognition. I., Bell System Technical Journal 40 (1961) 1–42.
[38] Erik Winfree, On the computational power of DNA annealing and ligation, DNA Based Computers (1996) 199–221.
[39] Erik Winfree, Algorithmic self-assembly of DNA, Ph.D. Thesis, California Insitute of Technology, Pasadena, CA, June 1998.
[40] Erik Winfree, Simulations of computing by self-assembly of DNA, Technical Report CS-TR:1998:22, California Insitute of Technology, Pasadena, CA, 1998.
[41] Erik Winfree, Self-healing tile sets, Nanotechnology: Science and Computation (2006) 55–78.
[42] Erik Winfree, Renat Bekbolatov, Proofreading tile sets: Error correction for algorithmic self-assembly, in: The 43rd Annual IEEE Symposium on Foundations of Computer Science, FOCS'02, vol. 2943, Madison, WI, June 2003, pp. 126–144.