Tight Bounds for Active Self-Assembly Using an Insertion Primitive *

Caleb Malchik, Andrew Winslow

Tufts University, Medford, MA 02155, USA.
caleb.malchik@tufts.edu, awinslow@cs.tufts.edu

Abstract. We prove two tight bounds on the behavior of a model of self-assembling particles introduced by Dabby and Chen (SODA 2012), called *insertion systems*, where monomers insert themselves into the middle of a growing linear polymer. First, we prove that the expressive power of these systems is equal to context-free grammars, answering a question posed by Dabby and Chen. Second, we prove that polymers of length $2^{\Theta(k^{3/2})}$ can be deterministically constructed by insertion systems of k monomer types in $O((\log n)^{5/3})$ expected time, and that this is the best possible in both the number of types and expected time.

Keywords: DNA computing, formal languages, polymers, context-free grammars

1 Introduction

In this work we study a theoretical model of *algorithmic self-assembly*, in which simple particles aggregate in a distributed manner to carry out complex functionality. Perhaps the the most well-studied theoretical model of algorithmic self-assembly is the *abstract Tile Assembly Model (aTAM)* of Winfree [15] consisting of square *tiles* irreversibly attach to a growing polyomino-shaped assembly according to matching edge colors. This model is capable of Turing-universal computation [15], self-simulation [5], and efficient assembly of general (scaled) shapes [14] and squares [1,13]. Despite this power, the model is incapable of assembling some shapes efficiently; a single row of n tiles requires n distinct tile types and $\Omega(n \log n)$ expected assembly time [2] and any shape with n tiles requires $\Omega(\sqrt{n})$ expected time to assemble [7].

Such a limitation may not seem so significant, except that a wide range of biological systems form complex assemblies in time polylogarithmic in the assembly site, as Dabby and Chen [4] and Woods et al. [16] observe. These biological systems are capable of such growth because their particles (e.g. living cells) *actively* carry out geometric reconfiguration. In the interest of both understanding naturally occurring biological systems and creating synthetic systems with additional capabilities, several models of *active self-assembly* have been proposed recently. These include the graph grammars of Klavins et al. [9,10], the *nubots* model

* A full version of this paper can be found at http://arxiv.org/abs/1401.0359

of Woods et al. [3,16], and the insertion systems of Dabby and Chen [4]. Both graph grammars and nubots are capable of a topologically rich set of assemblies and reconfigurations, but rely on stateful particles forming complex bond arrangments. In contrast, insertion systems consist of stateless particles forming a single chain of bonds. Indeed, all insertion systems are captured as a special case of nubots in which a linear polymer is assembled via parallel insertion-like reconfigurations, as in Theorem 5.1 of [17]. The simplicity of insertion systems makes their implementation in matter a more immediately attainable goal; Dabby and Chen [4] describe a direct implementation of these systems in DNA.

We are careful to make a distinction between *active self-assembly*, where assemblies undergo reconfiguration, and *active tile self-assembly* [6,8,11,12], where tile-based assemblies change their bond structure. Active self-assembly enables exponential assembly rates by enabling insertion of new particles throughout the assembly, while active tile self-assembly does not: assemblies formed consist of rigid tiles and the $\Omega(\sqrt{n})$ expected-time lower bound of Keenan, Schweller, Sherman, and Zhong [7] still applies.

1.1 Our results

We prove two tight bounds on the behavior of insertion systems. First, we consider what languages can be *expressed* by insertion systems, i.e. correspond to a set of polymers constructed by some insertion system. Dabby and Chen prove that only context-free languages are expressible by insertion systems, and ask whether every context-free language is indeed expressed by some insertion system. We answer this question in the affirmative, and as a consequence prove that the languages expressible by insertion systems are exactly the context-free languages.

Second, we consider constructing the largest finite polymers as fast as possible. Dabby and Chen prove that insertion systems with k monomer types can deterministically construct polymers of length $n = 2^{\Theta(\sqrt{k})}$ in $O(\log^3 n)$ expected time. We improve on both the polymer length *and* expected time by describing systems that deterministically constructing polymers of length $2^{\Theta(k^{3/2})}$ and $O((\log n)^{5/3})$ expected time by utilizing novel aspects of insertion systems. We also prove these systems are asymptotically optimal in both the length of the polymers they construct and the construction time.

2 Definitions

2.1 Grammars

A *context-free grammar* $\mathcal{G}$ is a 4-tuple $\mathcal{G} = (\Sigma, \Gamma, \Delta, S)$. The sets Σ and Γ are the *terminal* and *non-terminal symbols* of the grammar. The set Δ consists of *production rules* or simply *rules*, each of the form $L \to R_1 R_2 \cdots R_j$ with $L \in \Gamma$ and $R_i \in \Sigma \cup \Gamma$. Finally, the symbol $S \in \Gamma$ is a special *start symbol*. The *language of* $\mathcal{G}$, denoted $L(\mathcal{G})$, is the set of strings that can be *derived* by starting with S,

and repeatedly replacing a non-terminal symbol found on the left-hand side of some rule in Δ with the sequence of symbols on the right-hand side of the rule. The *size* of $\mathcal{G}$ is $|\Delta|$, the number of rules in $\mathcal{G}$. If every rule in Δ is of the form $L \to R_1 R_2$ or $L \to t$, with $R_1 R_2 \in \Gamma$ and $t \in \Sigma$, then the grammar is said to be in *Chomsky normal form*. Every context-free grammar can be converted to a grammar in Chomsky normal form while increasing the size of grammar by at most a factor of 2.

An *integer-pair grammar*, used in Section 3, is a context-free grammar in Chomsky normal form such that each non-terminal symbol is an integer pair (a, d), and each production rule has the form $(a, d) \to (a, b)(c, d)$ or $(a, d) \to t$.

2.2 Insertion systems

An *insertion system* in the active self-assembly model of Dabby and Chen [4] carries out the construction of a linear *polymer* consisting of constant length *monomers*. A polymer grows incrementally by the insertion of a monomer at an *insertion site* between two existing monomers in the polymer, according to complementary bonding sites between the monomer and the insertion site.

An insertion system $\mathcal{S}$ is defined as a 4-tuple $\mathcal{S} = (\Sigma, \Delta, Q, R)$. The first element, Σ, is a set of symbols. Each symbol $s \in \Sigma$ has a *complement* s^*. We denote the complement of a symbol s as $\overline{s}$, i.e. $\overline{s} = s^*$ and $\overline{s^*} = s$. The set Δ is a set of *monomer types*, each assigned a *concentration*. Each monomer is specified by a quadruple $(a, b, c, d)^+$ or $(a, b, c, d)^-$, where $a, b, c, d \in \Sigma \cup \{s^* : s \in \Sigma\}$, and each concentration is a real number between 0 and 1. The sum of all concentrations in Δ must be at most 1. The two symbols $Q = (a, b)$ and $R = (c, d)$ are special two-symbol monomers that together form the *initiator* of $\mathcal{S}$. It is required that either $\overline{a} = d$ or $\overline{b} = c$. The *size* of $\mathcal{S}$ is $|\Delta|$, the number of monomer types in $\mathcal{S}$.

A *polymer* is a sequence of monomers $Qm_1 m_2 \ldots m_n R$ where $m_i \in \Delta$ such that for each pair of adjacent monomers $(w, x, a, b)(c, d, y, z)$, either $\overline{a} = d$ or $\overline{b} = c$. The *length* of a polymer is the number of monomers, including Q and R, it contains. Each pair of adjacent monomer ends $(a, b)(c, d)$ form an *insertion site*. Monomers can be inserted into an insertion site $(a, b)(c, d)$ (and the sequence of monomers) according to the following rules (see Figure 1):

1. If $\overline{a} = d$, then any monomer $(\overline{b}, e, f, \overline{c})^+$ can be inserted.
2. If $\overline{b} = c$, then any monomer $(e, \overline{a}, \overline{d}, f)^-$ can be inserted.[1]

A monomer is inserted after time t, where t is an exponential random variable with rate equal to the concentration of the monomer type. The set of all polymers *constructed* by an insertion system is recursively defined as any polymer constructed by inserting a monomer into a polymer constructed by the system, beginning with the initiator. Note that the insertion rules guarantee by induction that for every insertion site $(a, b)(c, d)$, either $\overline{a} = d$ or $\overline{b} = c$.

[1] In [4], this rule is described as a monomer $(\overline{d}, f, e, \overline{a})^-$ that is inserted into the polymer as $(e, \overline{a}, \overline{d}, f)$.

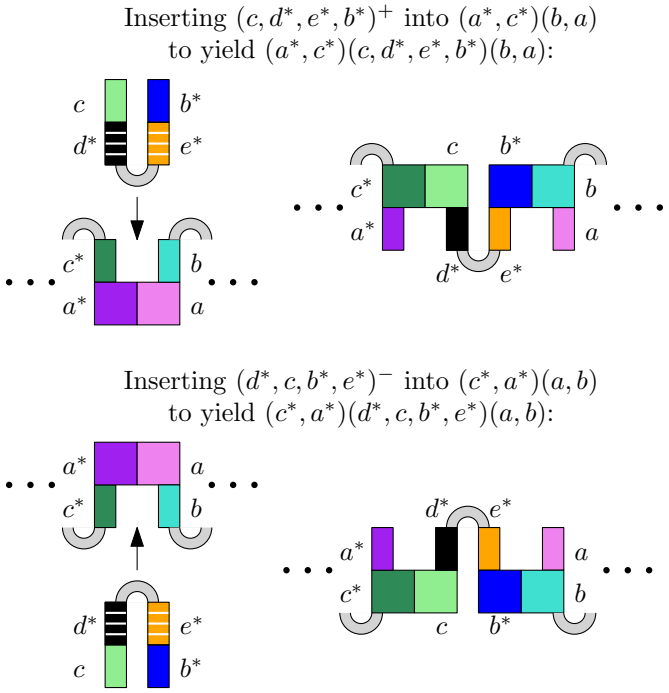

Fig. 1. A pictorial interpretation of the two insertion rules for monomers. Loosely based on Figure 2 and corresponding DNA-based implementation of [4].

We say that a polymer is *terminal* if no monomer can be inserted into any insertion site in the polymer, and that an insertion system *deterministically constructs* a polymer P if every polymer constructed by the system is either P or is non-terminal and has length less than that of P (i.e. can become P). The *stringification* of a polymer is the sequence of symbols in found on the polymer from left to right, e.g. $(a, b)(b^*, a, d, c)(c^*, a)$ has stringification abb^*adcc^*a. We call the set of stringifications of all terminal polymers of an insertion system $\mathcal{S}$ the *language* of $\mathcal{S}$, denoted $L(\mathcal{S})$.

2.3 Expressive power

Intuitively, a system *expresses* another if the terminal polymers or strings created by the system "look" like the terminal polymers or strings created by the other system. In the simplest instance, an integer-pair grammar $\mathcal{G}'$ is said to *express* a context-free grammar $\mathcal{G}$ if $L(\mathcal{G}') = L(\mathcal{G})$. Similarly, a grammar $\mathcal{G}$ is said to *express* an insertion system $\mathcal{S}$ if $L(\mathcal{S}) = L(\mathcal{G})$, i.e. if the set of stringifications of the terminal polymers of $\mathcal{S}$ equals the language of $\mathcal{G}$.

An insertion system $\mathcal{S} = (\Sigma', \Delta', Q', R')$ is said to express a grammar $\mathcal{G} = (\Sigma, \Gamma, \Delta, S)$ if there exists a function $g : \Sigma' \cup \{s^* : s \in \Sigma'\} \to \Sigma \cup \{\varepsilon\}$ such that $\{g(s'_1)g(s'_2)\dots g(s'_n) : s'_1 s'_2 \dots s'_n \in L(\mathcal{S})\} = L(\mathcal{G})$. More precisely, we require

that there exists a fixed integer κ such that for any substring $s'_{i+1}s'_{i+2}\ldots s'_{i+\kappa}$ in a string in $L(\mathcal{S})$, $\{g(s'_{i+1}), g(s'_{i+2}), \ldots, g(s'_{i+\kappa})\} \neq \{\varepsilon\}$. That is, the insertion system symbols mapping to grammar terminal symbols are evenly distributed throughout the polymer. The requirement of a fixed integer κ prevents the possibility of a polymer containing arbitrarily long and irregular regions of "garbage" monomers.

3 The Expressive Power of Insertion Systems

Dabby and Chen proved that any insertion system has a context-free grammar expressing it. They construct such a grammar by creating a non-terminal for every possible pair of adjacent monomer types, and a production rule with this left-hand side non-terminal for each monomer that can be inserted into the insertion site formed by this pair. Here we give a reduction in the other direction, resolving, in the affirmative, the question posed by Dabby and Chen of whether context-free grammars and insertion systems have the same expressive power:

Theorem 1. *For every context-free grammar G, there exists an insertion system that expresses G.*

The primary difficulty in proving Theorem 1 lies in developing a way to simulate the "complete" replacement that occurs during derivation with the "incomplete" replacement that occurs when an insertion site is inserted into. For instance, *bcAbc* becomes *bcDDbc* via a production rule $A \to DD$ and A is completely replaced by DD. On the other hand, inserting a monomer $(b^*, d, d, c)^+$ into a site $(a, b)(c^*, a^*)$ yields the consecutive sites $(a, b)(b^*, d)$ and $(d, c)(c^*, a^*)$, with $(a, b)(c^*, a^*)$ only partially replaced – the left side of the first site and the right side of second site together form the initial site. This behavior constrains how replacement can be captured by insertion sites, and the κ parameter of the definition of expression (Section 2.3) prevents eliminating the issue via additional insertions.

We overcome this difficulty by proving Theorem 1 in two steps. First, we prove that integer-pair grammars, a constrained type of grammar with incomplete replacements, are able to express context-free grammars (Lemma 1). Second, we prove integer-pair grammars can be expressed by insertion systems (Lemma 2).

Lemma 1. *For every context-free grammar $\mathcal{G}$, there exists an integer-pair grammar that expresses $\mathcal{G}$.*

Lemma 2. *For every integer-pair grammar $\mathcal{G}$, there exists an insertion system that expresses $\mathcal{G}$.*

Proof. Let $\mathcal{G} = (\Sigma, \Gamma, \Delta, S)$. The integer-pair grammar $\mathcal{G}$ is expressed by an insertion system $\mathcal{S} = (\Sigma', \Delta', Q', R')$ that we now define. Let $\Sigma' = \{s_a, s_b : (a, b) \in \Gamma\} \cup \{u, x\} \cup \Sigma$. Let $\Delta' = \Delta'_1 \cup \Delta'_2 \cup \Delta'_3 \cup \Delta'_4$, where

$$\Delta'_1 = \{(s_b, u, s_b^*, x)^- : (a, d) \to (a, b)(c, d) \in \Delta\}$$

$$\Delta_2' = \{(s_a, s_b, s_c^*, s_d^*)^+ : (a,d) \to (a,b)(c,d) \in \Delta\}$$
$$\Delta_3' = \{(x, s_c, u^*, s_c^*)^- : (a,d) \to (a,b)(c,d) \in \Delta\}$$
$$\Delta_4' = \{(s_a, t, x, s_d^*)^+ : (a,d) \to t \in \Delta\}$$

We give each monomer type equal concentration, although the precise concentrations are not important for expressive power. Let $Q' = (u^*, a^*)$ and $R' = (b, u)$, where $S = (a, b)$.

Insertion types. We start by proving that for any polymer constructed by $\mathcal{S}$, only the following types of insertions of a monomer m_2 between two adjacent monomers $m_1 m_3$ are possible:

1. $m_1 \in \Delta_2'$, $m_2 \in \Delta_3'$, $m_3 \in \Delta_1'$
2. $m_1 \in \Delta_3'$, $m_2 \in \Delta_2' \cup \Delta_4'$, $m_3 \in \Delta_1'$
3. $m_1 \in \Delta_3'$, $m_2 \in \Delta_1'$, $m_3 \in \Delta_2'$

Moreover, we claim that for every adjacent $m_1 m_3$ pair satisfying one of these conditions, an insertion *is* possible. That is, there is a monomer m_2 that can be inserted, necessarily from the monomer subset specified.

Consider each possible combination of $m_1 \in \Delta_i'$ and $m_3 \in \Delta_j'$, respectively, with $i, j \in \{1, 2, 3, 4\}$. Observe that for an insertion to occur at insertion site $(a, b)(c, d)$, the symbols $\overline{a}$, $\overline{b}$, $\overline{c}$, and $\overline{d}$ must each occur on some monomer. Then since x^* and t^* do not appear on any monomers, any i, j with $i \in \{1, 4\}$ or $j \in \{3, 4\}$ cannot occur. This leaves monomer pairs (Δ_i', Δ_j') with $(i, j) \in \{(2, 1), (2, 2), (3, 1), (3, 2)\}$.

Insertion sites between (Δ_2', Δ_1') pairs have the form $(s_c^*, s_d^*)(s_b, u)$, so an inserted monomer must have the form $(s_e, s_c, s_u^*, s_f)^-$ and is in Δ_3'. An insertion site $(s_c^*, s_d^*)(s_b, u)$ implies a rule of the form $(e, d) \to (e, f)(c, d)$ in Δ, so there exists a monomer $(x, s_c, u^*, s_c^*)^- \in \Delta_3'$ that can be inserted.

Insertion sites between (Δ_3', Δ_2') pairs have the form $(u^*, s_c^*)(s_a, s_b)$, so an inserted monomer must have the form $(\_, u, s_b^*, \_)^-$ and thus is in Δ_1'. An insertion site $(u^*, s_c^*)(s_a, s_b)$ implies a rule of the form $(a, d) \to (a, b)(e, d)$ in Γ, so there exists a monomer $(s_b, u, s_b^*, x)^- \in \Delta_1'$ that can be inserted.

Insertion sites between (Δ_2', Δ_2') pairs can only occur once a monomer $m_2 \in \Delta_2'$ has been inserted between a pair of adjacent monomers $m_1 m_3$ with either $m_1 \in \Delta_2'$ or $m_3 \in \Delta_2'$, but not both. But we just proved that all such such possible insertions only permit $m_2 \in \Delta_3' \cup \Delta_1'$. Moreover, the initial insertion site between Q' and R' has the form $(u^*, s_a^*)(s_b, u)$ of an insertion site with $m_1 \in \Delta_3'$ and $m_3 \in \Delta_1'$. So no pair of adjacent monomers $m_1 m_3$ are ever both from Δ_2' and no insertion site between (Δ_2', Δ_2') pairs can ever exist.

Insertion sites between (Δ_3', Δ_1') pairs have the form $(u^*, s_c^*)(s_b, u)$, so an inserted monomer must have the form $(s_c, \_, \_, b^*)^+$ or $(\_, u, u^*, \_)^-$ and is in Δ_2' or Δ_4'. We show by induction that for each such insertion site $(u^*, s_c^*)(s_b, u)$ that $(c, b) \in \Gamma$. First, observe that this is true for the insertion site $(u^*, s_a^*)(s_b, u)$ between Q' and R', since $(a, b) = S \in \Gamma$. Next, suppose this is true for all insertion sites of some polymer and a monomer $m_2 \in \Delta_2' \cup \Delta_4'$ is about to be inserted into the polymer between monomers from Δ_3' and Δ_1'. Inserting a

monomer $m_2 \in \Delta_4'$ only reduces the set of insertion sites between monomers in Δ_3' and Δ_1', and the inductive hypothesis holds. Inserting a monomer $m_2 \in \Delta_2'$ induces new (Δ_3', Δ_2') and (Δ_2', Δ_1') insertion site pairs between m_1m_2 and m_2m_3. These pairs must accept two monomers $m_4 \in \Delta_1$ and $m_5 \in \Delta_3$, inducing a sequence of monomers $m_1m_4m_2m_5m_3$ with adjacent pairs (Δ_3', Δ_1'), (Δ_1', Δ_2'), (Δ_2', Δ_3'), (Δ_3', Δ_1'). Only the first and last pairs permit insertion and both are (Δ_3', Δ_1') pairs.

Now consider the details of the three insertions yielding $m_1m_4m_2m_5m_3$, starting with m_1m_3. The initial insertion site m_1m_3 must have the form $(u^*, s_a^*)(s_d, u)$. So the sequence of insertions has the following form, with the last two insertions interchangeable. The symbol $\diamond$ is used to indicate the site being modified and the inserted monomer shown in bold:

$$(u^*, s_a^*) \diamond (s_d, u)$$
$$(u^*, s_a^*) \diamond (\boldsymbol{s_a, s_b, s_c^*, s_d^*})(s_d, u)$$
$$(u^*, s_a^*)(\boldsymbol{s_b, u, s_b^*, x})(s_a, s_b, s_c^*, s_d^*) \diamond (s_d, u)$$
$$(u^*, s_a^*)(s_b, u, s_b^*, x)(s_a, s_b, s_c^*, s_d^*)(\boldsymbol{x, s_c, u^*, s_c^*})(s_d, u)$$

The two resulting (Δ_3', Δ_1') pair insertion sites are $(u^*, s_a^*)(s_b, u)$ and $(u^*, s_c^*)(s_d, u)$. Assume, by induction, that the monomer m_2 must exist. So there is a rule $(a, d) \to (a, b)(c, d) \in \Delta$ and $(a, b), (c, d) \in \Gamma$, fulfilling the inductive hypothesis. So for every insertion site $(u^*, s_c^*)(s_b, u)$ between a (Δ_3', Δ_1') pair there exists a non-terminal $(c, b) \in \Gamma$. So for every adjacent monomer pair m_1m_3 with $m_1 \in \Delta_3'$ and $m_3 \in \Delta_1'$, there exists a monomer $m_2 \in \Delta_2' \cup \Delta_4'$ that can be inserted between m_1 and m_2.

Partial derivations and terminal polymers. Next, consider the sequence of insertion sites between (Δ_3', Δ_1') pairs in a polymer constructed by a modified version of $\mathcal{S}$ lacking the monomers of Δ_4'. We claim that there is a constructed polymer with a sequence $(u^*, s_{a_1}^*)(s_{b_1}, u), (u^*, s_{a_2}^*)(s_{b_2}, u), \ldots, (u^*, s_{a_i}^*)(s_{b_i}, u)$ of these insertion sites if and only if there is a partial derivation $(a_1, b_1)(a_2, b_2) \ldots (a_i, b_i)$ of a string in $L(\mathcal{G})$. This follows directly from the previous proof by observing that two new adjacent (Δ_3', Δ_1') pair insertion sites $(u^*, s_a^*)(s_b, u)$ and $(u^*, s_c^*)(s_d, u)$ can replace a (Δ_3', Δ_1') pair insertion site if and only if there exists a rule $(a, d) \to (a, b)(c, d) \in \Delta$.

Observe that any string in $L(\mathcal{G})$ can be derived by first deriving a partial derivation containing only non-terminals, then applying only rules of the form $(a, d) \to t$. Similarly, since the monomers of Δ_4' never form half of a valid insertion site, any terminal polymer of $\mathcal{S}$ can be constructed by first generating a polymer containing only monomers in $\Delta_1' \cup \Delta_2' \cup \Delta_3'$, then only inserting monomers from Δ_4'. Also note that the types of insertions possible in $\mathcal{S}$ imply that in any terminal polymer, any triple of adjacent monomers $m_1m_2m_3$ with $m_1 \in \Delta_i'$, $m_2 \in \Delta_j'$, and $m_3 \in \Delta_k'$, that $(i, j, k) \in \{(4, 1, 2), (1, 2, 3), (2, 3, 4), (3, 4, 1)\}$, with the first and last monomers of the polymer in Δ_4'.

Expression. Define the following piecewise function $g : \Sigma' \cup \{s^* : s \in \Sigma'\} \to \Sigma \cup \{\varepsilon\}$ that maps to ε except for the second symbols of monomers in Δ_4'.

$$g(s) = \begin{cases} t, \text{ if } t \in \Sigma \\ \varepsilon, \text{ otherwise} \end{cases}$$

Observe that every string in $L(\mathcal{S})$ has length $2 + 4 \cdot (4n-3) + 2 = 16n - 8$ for some $n \geq 0$. Also, for each string $s'_1 s'_2 \dots s'_{16n-8} \in L(\mathcal{S})$, $g(s'_1)g(s'_2)\dots g(s'_{16n-8}) = \varepsilon^3 t_1 \varepsilon^{16} t_2 \varepsilon^{16} \dots t_n \varepsilon^5$. There is a terminal polymer with stringification in $L(\mathcal{S})$ yielding the sequence $s_1 s_2 \dots s_n$ if and only if the polymer can be constructed by first generating a terminal polymer excluding Δ'_4 monomers with a sequence of (Δ'_3, Δ'_1) insertion pairs $(a_1, b_1)(a_2, b_2)\dots(a_n, b_n)$ followed by a sequence of insertions of monomers from Δ'_4 with second symbols $t_1 t_2 \dots t_n$. Such a generation is possible if and only if $(a_1, b_1)(a_2, b_2)\dots(a_n, b_n)$ is a partial derivation of a string in $L(\mathcal{G})$ and $(a_1, b_1) \to t_1, (a_2, b_2) \to t_2, \dots, (a_n, b_n) \to t_n \in \Delta$. So applying the function g to the stringifications of the terminal polymers of $\mathcal{S}$ gives $L(\mathcal{G})$, i.e. $L(\mathcal{S}) = L(\mathcal{G})$. Moreover, the second symbol in every fourth monomer in a terminal polymer of $\mathcal{S}$ maps to a symbol of Σ using g. So $\mathcal{S}$ expresses $\mathcal{G}$ with the function g and $\kappa = 16$. □

4 Positive Results for Polymer Growth

Dabby and Chen also consider the size and speed of constructing finite polymers. They give a construction achieving the following result:

Theorem 2 ([4]). *For any positive integer r, there exists an insertion system with $O(r^2)$ monomer types that deterministically constructs a polymer of length $n = 2^{\Theta(r)}$ in $O(\log^3 n)$ expected time.*

We improve on this construction significantly in both polymer length and expected running time. In Section 5 we prove that our construction is the best possible with respect to both the polymer length and construction time.

Theorem 3. *For any positive integer r, there exists an insertion system with $O(r^2)$ monomer types that deterministically constructs a polymer of length $n = 2^{\Theta(r^3)}$ in $O((\log n)^{5/3})$ expected time.*

Proof. We give a constructive proof. The approach is to implement a three variable counter where each variable ranges over the values 0 to r, effectively carrying out the execution of a triple for-loop. Insertion sites of the form $(s_a, s_b)(s_c, s^*_a)$ are used to encode the state of the counter, where a, b, and c are the variables of the outer, inner, and middle loops, respectively.

1. (Inner): If $0 \leq b < r$, then $(s_a, s_b)(s_c, s^*_a)$ becomes $(s_a, s_{b+1})(s_c, s^*_a)$.
2. (Middle): If $b = r$ and $0 \leq c < r$, then $(s_a, s_b)(s_c, s^*_a)$ becomes $(s_a, s_0)(s_{c+1}, s^*_a)$.
3. (Outer): If $b = c = r$ and $0 \leq a < r$, then $(s_a, s_b)(s_c, s^*_a)$ becomes $(s_{a+1}, s_0)(s_0, s^*_{a+1})$.

A site is *modified* by a sequence of monomer insertions that yields a new usable site where all other sites created by the insertion sequence are unusable.

For instance, we modify a site $(s_a, \boldsymbol{s_b})(s_c, s_a^*)$ to become $(s_a, \boldsymbol{s_d})(s_c, s_a^*)$, written $(s_a, s_b)(s_c, s_a^*) \to (s_a, s_d)(s_c, s_a^*)$, by adding the monomer types $(s_b^*, x, u, s_c^*)^+$ and $(x, u^*, s_a, s_b)^-$ to the system, where x is a special symbol whose complement is not found on any monomer. These two monomer types cause the following sequence of insertions, using $\diamond$ to indicate the site being modified and the inserted monomer shown in bold:

$$(s_a, s_b) \diamond (s_c, s_a^*)$$

$$(s_a, s_b)\boldsymbol{(s_b^*, x, u, s_c^*)} \diamond (s_c, s_a^*)$$

$$(s_a, s_b)(s_b^*, x, u, s_c^*)\boldsymbol{(x, u^*, s_a, s_d)} \diamond (s_c, s_a^*)$$

We call this simple modification, where a single symbol in the insertion site is replaced with another symbol, a *replacement*. Four types of replacements, seen in Table 1, can each be implemented by a pair of corresponding monomers.

Replacement	Monomers
$(s_a, \boldsymbol{s_b})(s_c, s_a^*) \to (s_a, \boldsymbol{s_d})(s_c, s_a^*)$	$(s_b^*, x, u, s_c^*)^+$, $(x, u^*, s_a, s_d)^-$
$(s_a, s_b)(\boldsymbol{s_c}, s_a^*) \to (s_a, s_b)(\boldsymbol{s_d}, s_a^*)$	$(s_b^*, u, x, s_c^*)^+$, $(s_d, s_a^*, u^*, x)^-$
$(\boldsymbol{s_b}, s_a)(s_a^*, s_c) \to (\boldsymbol{s_d}, s_a)(s_a^*, s_c)$	$(x, s_b^*, s_c^*, u)^-$, $(u^*, x, s_d, s_a)^+$
$(s_b, s_a)(s_a^*, \boldsymbol{s_c}) \to (s_b, s_a)(s_a^*, \boldsymbol{s_d})$	$(u, s_b^*, s_c^*, x)^-$, $(s_a^*, s_d, x, u^*)^+$

Table 1. The four types of replacement steps and monomer pairs that implement them. The symbol u can be any symbol, and x is a special symbol whose complement does not appear on any monomer.

Each of the three increment types are implemented using a sequence of site modifications. The resulting triple for-loop carries out a sequence of $\Theta(r^3)$ insertions, constructing a $\Theta(r^3)$-length polymer. A $2^{\Theta(r^3)}$-length polymer is achieved by simultaneously duplicating each site during each inner increment. Because the for-loop runs for $\Theta(r^3)$ steps and duplicates at a constant fraction of these steps (those with $0 \leq b < r$), the number of counters reaching the final $a = b = c = r$ state is $2^{\Theta(r^3)}$. In the remainder of the proof, we detail the implementation of each increment type, starting with the simplest: middle increments.

Middle increment. A middle increment of a site $(s_a, s_b)(s_c, s_a^*)$ occurs when the site has the form $(s_a, s_r)(s_c, s_a^*)$ with $0 \leq c < r$, performing the modification $(s_a, s_r)(s_c, s_a^*) \to (s_a, s_0)(s_{c+1}, s_a^*)$. We implement middle increments using a sequence of three replacements:

$$(s_a, s_r)(s_c, s_a^*) \xrightarrow{1} (s_a, s_r)(s_{f_1(c)}, s_a^*) \xrightarrow{2} (s_a, s_0)(s_{f_1(c)}, s_a^*) \xrightarrow{3} (s_a, s_0)(s_{c+1}, s_a^*)$$

where $f_i(n) = n + 2ir^2$. The use of f is to avoid unintended interactions between monomers, since for any n_1, n_2 with $0 \leq n_1, n_2 \leq r$, $f_i(n_1) \neq f_j(n_2)$ for all $i \neq j$. Compiling this sequence of replacements into monomer types gives the following set:

1. (Step 1): $(s_r^*, s_{f_2(c)}, x, s_c^*)^+$ and $(s_{f_1(c)}, s_a^*, s_{f_2(c)}^*, x)^-$.
2. (Step 2): $(s_r^*, x, s_{f_3(c)}, s_{f_1(c)}^*)^+$ and $(x, s_{f_3(c)}^*, s_a, s_0)^-$.
3. (Step 3): $(s_0^*, s_{f_4(c+1)}, x, s_{f_1(c)}^*)^+$ and $(s_{c+1}, s_a^*, s_{f_4(c+1)}^*, x)^-$.

Since each inserted monomer has an instance of x, all other insertion sites created are unusable. This is true of the insertions used for outer increments and duplications as well.

Outer increment. An outer increment of the site $(s_a, s_b)(s_c, s_a^*)$ occurs when the site has the form $(s_a, s_r)(s_r, s_a^*)$ with $0 \leq a < r$. We implement this step using a two-phase sequence of three (regular) replacements and a special quadruple replacement (Step 3):

$$(s_a, s_r)(s_r, s_a^*) \xrightarrow{1} (s_a, s_{f_5(a)})(s_r, s_a^*) \xrightarrow{2} (s_a, s_{f_5(a)})(s_{f_5(a)}^*, s_a^*)$$

$$(s_a, s_{f_5(a)})(s_{f_5(a)}^*, s_a^*) \xrightarrow{3} (s_{a+1}, s_{f_5(0)})(s_0, s_{a+1}^*) \xrightarrow{4} (s_{a+1}, s_0)(s_0, s_{a+1}^*)$$

At each step, a (necessary) complementary pair of symbols is maintained, which results in a sequence of more than 4 replacements. As with inner and middle increments, we compile replacement steps 1, 2, and 4 into monomers using Table 1. Step 3 is a special pair of monomers.

1. (Step 1): $(s_r^*, x, s_{f_6(r)}, s_r^*)^+$ and $(x, s_{f_6(r)}^*, s_a, s_{f_5(a)})^-$.
2. (Step 2): $(s_{f_5(a)}^*, s_{f_7(r)}^*, x, s_r^*)^+$ and $(s_{f_5(a)}^*, s_a^*, s_{f_7(r)}, x)^-$.
3. (Step 3): $(s_{f_5(a)}^*, x, s_{a+1}, s_{f_5(a)})^+$ and $(s_0, s_{a+1}^*, s_a, x)^-$.
4. (Step 4): $(s_{f_5(a)}^*, x, s_{f_7(r)}, s_0^*)^+$ and $(x, s_{f_7(r)}^*, s_{a+1}, s_0)^-$.

Inner increment. The inner increment has two phases. The first phase performs the modification $(s_a, s_b)(s_c, s_a^*) \to (s_a, s_b)(s_{f_8(c)}, s_a^*) \ldots (s_a, s_{b+1})(s_c, s_a^*)$, yielding an incremented version of the original site and one other site. The second phase is $(s_a, s_b)(s_{f_8(c)}, s_a^*) \to (s_a, s_{b+1})(s_c, a^*)$, transforming the second site into an incremented version of the original site.

For the first phase, we use the three monomers $(s_b^*, s_{f_8(c)}, s_{f_8(b+1)}, s_c^*)^+$, $(s_{f_8(c)}, s_a^*, s_{f_8(c)}^*, x)^-$, and $(x, s_{f_8(b+1)}^*, s_a, s_{b+1})^-$ and call the entire phase Step 1. The site $(s_a, s_b)(s_{f_8(c)}, s_a^*)$ is transformed into $(s_a, s_{b+1})(s_c, s_a^*)$ by a sequence of replacement steps:

$$(s_a, s_b)(s_{f_8(c)}, s_a^*) \xrightarrow{2} (s_a, s_{f_9(b)})(s_{f_8(c)}, s_a^*) \xrightarrow{3} (s_a, s_{f_9(b)})(s_c, s_a^*) \xrightarrow{4} (s_a, s_{b+1})(s_c, s_a^*)$$

As with previous sequences of replacement steps, we compile this sequence into a set of monomers:

1. (Step 2): $(s_b^*, x, s_{f_{10}(b)}, s_{f_8(c)}^*)^+$ and $(x, s_{f_{10}(b)}^*, s_a, s_{f_9(b)})^-$.
2. (Step 3): $(s_{f_9(b)}^*, s_{f_{11}(c)}, x, s_{f_8(c)}^*)^+$ and $(s_c, s_a^*, s_{f_{11}(c)}^*, x)^-$.
3. (Step 4): $(s_{f_9(b)}^*, x, s_{f_{12}(b+1)}, s_c^*)^+$ and $(x, s_{f_{12}(b+1)}^*, s_a, s_{b+1})^-$.

When combined, the two phases of duplication modify $(s_a, s_b)(s_c, s_a^*)$ to become $(s_a, s_{b+1})(s_c, s_a^*) \ldots (s_a, s_{b+1})(s_c, s_a^*)$, where all sites between the duplicated sites are unusable.

Putting it together. The system starts with the intiator $(s_0, s_0)(s_0, s_0^*)$. Each increment of the counter occurs either through a middle increment, outer increment, or a duplication. There are at most $(r+1)^2$ monomer types in each family and $O(r^2)$ monomer types total. The size P_i of a subpolymer with an initiator encoding some value i between 0 and $(r+1)^3 - 1$ can be bounded by $2P_{i+2} + 9 \leq P_i 2P_{i+1} + 9$ with $P_{(r+1)^3-2} > 0$. So P_0, the size of the terminal polymer, is $2^{\Theta(r^3)}$.

Running time. Define the concentration of each monomer type to be equal. There are less than $39r^2$ monomer types, so each monomer type has concentration at least $1/(39r^2)$. The polymer is complete as soon as every insertion site has been modified to be $(r, r)(r, r^*)$ and the monomer $(s_r^*, x, s_{f_7(r)}, s_r^*)^+$ has been inserted. There are fewer than 2^{8r^3} such insertions, and each insertion can occur once at most $9 \cdot 8r^3 = 72r^3$ previous insertions have occurred. So an upper bound on the expected time T_r for each such insertion is described as a sum of $72r^3$ random variables, each with expected time $39r^2$. The Chernoff bound for exponential random variables implies $\text{Prob}[T_r > 39r^2 \cdot 72r^3(1+\delta)] \leq e^{-r^5\delta/2}$ for all $\delta \geq 2$ and $T_{\mathcal{S}_r}$, the total running time of the system, has $\text{Prob}[T_{\mathcal{S}_r} > 39r^2 \cdot 72r^3(1+\delta)] \leq 2^{-r^5\delta/4}$ for all $\delta \geq 32$. So the expected value of $T_{\mathcal{S}_r}$, the construction time, is $O(r^5) = O((\log n)^{5/3})$ with an exponentially decaying tail probability. □

5 Negative Results for Polymer Growth

Here we prove that our system constructs polymers using an optimal number of monomer types and in optimal expected time.

Theorem 4. *Any polymer deterministically constructed by an insertion system with k monomer types has length $2^{O(k^{3/2})}$.*

Theorem 5. *Deterministically constructing a polymer of length n takes $\Omega((\log n)^{5/3})$ expected time.*

Acknowledgments

We wish to thank Benjamin Hescott for many helpful discussions and anonymous reviewers for feedback that helped to improve the presentation of the paper.

References

1. L. Adleman, Q. Cheng, A. Goel, and M.-D. Huang. Running time and program size for self-assembled squares. In *Proceedings of 33rd ACM Symposium on Theory of Computing (STOC)*, 2001.

2. L. M. Adleman, Q. Cheng, A. Goel, M.-D. Huang, and H. Wasserman. Linear self-assemblies: equilibria, entropy and convergence rates. In *Proceedings of 6th International Conference on Difference Equations and Applications*, 2001.
3. M. Chen, D. Xin, and D. Woods. Parallel computation using active self-assembly. In D. Soloveichik and B. Yurke, editors, *DNA Computing and Molecular Programming*, volume 8141 of *LNCS*, pages 16–30. 2013.
4. N. Dabby and H.-L. Chen. Active self-assembly of simple units using an insertion primitive. In *Proceedings of 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1526–1536, 2012.
5. D. Doty, J. H. Lutz, M. J. Patitz, R. T. Schweller, S. M. Summers, and D. Woods. The tile assembly model is intrinsically universal. In *Proceedings of 53rd IEEE Symposium on Foundations of Computer Sciences (FOCS)*, pages 302–310, 2012.
6. J. Hendricks, J. E. Padilla, M. J. Patitz, and T. A. Rogers. Signal transmission across tile assemblies: 3d static tiles simulate active self-assembly by 2d signal-passing tiles. In D. Soloveichik and B. Yurke, editors, *DNA Computing and Molecular Programming*, volume 8141 of *LNCS*, pages 90–104. Springer Berlin Heidelberg, 2013.
7. A. Keenan, R. Schweller, M. Sherman, and X. Zhong. Fast arithmetic in algorithmic self-assembly. Technical report, arXiv, 2013.
8. A. Keenan, R. Schweller, and X. Zhong. Exponential replication of patterns in the signal tile assembly model. In D. Soloveichik and B. Yurke, editors, *DNA Computing and Molecular Programming*, volume 8141 of *LNCS*, pages 118–132. Springer Berlin Heidelberg, 2013.
9. E. Klavins. Universal self-replication using graph grammars. In *Proceedings of International Conference on MEMS, NANO, and Smart Systems*, pages 198–204, 2004.
10. E. Klavins, R. Ghrist, and D. Lipsky. Graph grammars for self assembling robotic systems. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 5, pages 5293–5300, 2004.
11. U. Majumder, T. H. LaBean, and J. H. Reif. Activatable tiles: Compact, robust programmable assembly and other applications. In M. H. Garzon and H. Yan, editors, *DNA Computing and Molecular Programming*, volume 4848 of *LNCS*, pages 15–25. Springer Berlin Heidelberg, 2008.
12. J. Padilla, M. J. Patitz, R. Pena, R. T. Schweller, N. C. Seeman, R. Sheline, S. M. Summers, and X. Zhong. Asynchronous signal passing for tile self-assembly: fuel efficient computation and efficient assembly of shapes. In G. Mauri, A. Dennunzio, L. Manzoni, and A. E. Porreca, editors, *Unconventional Comp. and Natural Comp. (UCNC)*, volume 7956 of *LNCS*, pages 174–185. Springer Berlin Heidelberg, 2013.
13. P. W. K. Rothemund and E. Winfree. The program-size complexity of self-assembled squares (extended abstract). In *Proceedings of 32nd ACM Symposium on Theory of Computing (STOC)*, pages 459–468, 2000.
14. D. Soloveichik and E. Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007.
15. E. Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, Caltech, 1998.
16. D. Woods, H.-L. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proceedings of 4th Conference on Innovations in Theoretical Compuer Science (ITCS)*, pages 353–354, 2013.
17. D. Woods, H.-L. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. Technical report, arXiv, 2013.