Graph-theoretic formalization of hybridization in DNA sticker complexes

Robert Brijder, Joris J.M. Gillis*, and Jan Van den Bussche

Hasselt University and transnational University of Limburg

Abstract. Sticker complexes are a formal graph-based data model for a restricted class of DNA complexes, motivated by potential applications to databases. This data model allows for a purely declarative definition of hybridization. We introduce the notion of terminating hybridization, and characterize this notion in purely graph-theoretic terms. Terminating hybridization can still produce results of exponential size. We indicate a class of complexes where hybridization is guaranteed to be polynomially bounded.

1 Introduction

Since Adleman's experiment [2], DNA Computing has greatly evolved, and many different modes of computation have been invented and investigated [3, 21, 6, 25, 34, 26, 29, 14, 5, 33, 30, 32, 22]. A major goal throughout this evolution has been to achieve autonomy of computation, and indeed this is a highly desirable feature of computation in general.

At the same time, DNA Computing has also high potential for database applications [4, 10, 35, 24]. Indeed, the nanoscale and relative indestructibility of single DNA strands are very promising properties for database storage. Moreover, the highly parallel mode of operation that can be achieved in DNA Computing is a nice match with the bulk-processing nature of database computations.

Autonomy of computation is perhaps less crucial for databases, where indeed traditionally a strict line is drawn between the data, and the query or update operations performed on the data [15]. Also, in database theory [1], one expects formal data models defined on the logical level, and declarative definitions of the basic data manipulation operations.

In the present paper, in the context of a formal data model of DNA complexes, we focus on hybridization, one of the cornerstone operations in DNA computing. The data model is that of *sticker complexes*, a graph-theoretically defined formalization of DNA complexes of a limited format. Sticker complexes have been shown in an earlier paper [16] to be adequate for database computations in DNA. Indeed, while it is relatively straightforward to represent relational databases in DNA, a good data model for database computation must also be able to represent all intermediate data structures needed to support database

* Ph.D. Fellow of the Research Foundation Flanders (FWO).

operations. Specifically, it has been shown that sticker complexes are adequate to support a complete simulation of the operations of the relational algebra, which provides a set of core operations in relational databases [15]. The intermediate data structures involved in the simulation of the relational algebra are quite complex as they need to support the creation of circular strands.

The problem addressed in the present paper is to understand the well-definedness and *termination* of the hybridization operation on sticker complexes. Here we are considering hybridization as a database operation, like the Cartesian product (related to the relational join). When we want to construct the Cartesian product $U \times V$ of two sets U and V, with U of size m and V of size n, we need in principle n copies of every element of U, and m copies of every element of V, so that we have enough "material" to construct the $m \times n$-element set $\{(u, v) \mid u \in U \;\&\; v \in V\}$. When more copies are provided of some elements of U or V, some duplicate pairs can be constructed, but no really new information is generated. When hybridization has this behavior, we say it *terminates.*

The main result of this paper is to provide a purely graph-theoretic characterization of termination of hybridization, which will also imply that termination is decidable for sticker complexes. This result emphasizes the restricted nature of the sticker complex data model, since it is well known that termination is undecidable for Turing-universal computation models [18]. The investigation of computation models that are not computationally complete, and the corresponding search for the right balance between sufficient expressive power and low complexity, is one of the hallmarks of database theory [1].

We also investigate complexity issues related to DNA hybridization. Even when hybridization in a given DNA complex terminates, depending on the structure of the complex, an exponential amount of material may be required to produce the complete result. This problem was already present in Adleman's solution to the Hamiltonian Path problem [17], and we show it can still occur within the limited context of sticker complexes. Since such exponential behavior is undesirable, and also not needed to support typical database operations, we would like to avoid it.

We will show that the result of hybridization splits up, graph-theoretically, in a number of connected components, and each component is polynomial in size. Hence, the exponentiality is confined to the possible number of distinct components. Furthermore, we identify a broad family of classes of DNA complexes, called *c-bounded complexes*, within which hybridization is guaranteed to require only a polynomial amount of resources.

2 Related work

In one of the first papers on DNA computing, Reif already defined a formal data structure of DNA complexes [23]. Our data structures are simpler in an effort to avoid unrealistic or otherwise complicated and unmanageable secondary structures. (Reif avoids these by invoking an oracle for feasibility.) Our simplification is that single strands are either all-positive or all-negative, and moreover, negative strands have length at most two. The short negative strands can be thought of as stickers; thus the name "sticker complexes". Our previous work showed that the restrictions of sticker complexes do not preclude interesting database computations. An important feature of our model, which is lacking in Reif's, is the formal distinction between the structural content of a complex, and the complex as used in reactions, with multiples of each connected component present in surplus quantities.

The use of short stickers in DNA computing originates with Roweis et al. [26], where stickers were used to turn bits on or off. We use stickers to bind strands together so that possibly complex secondary structures are formed.

The present work also fits in a recent trend of integrating formal methods (such as process calculi in computational systems biology [7]) with DNA computing [8, 20]. Yet the formalisms we use are different from process calculi and comprise mainly set theory, graph theory, and logic-based query languages. The computational power of hybridization in various models of formal languages has been intensively studied, e.g., [21, 34].

Last but not least, our formal model of hybridization is strikingly comparable in spirit to the model of Jonoska, McColm and Staninska [19]. That paper introduces the notion of a "pot type"; for the purpose of hybridization, the model of pot types and our model of sticker complexes are roughly equivalent. It was shown that *weak satisfiability* is decidable in polynomial time. Using our own terminology as introduced in Section 4, a complex is weakly satisfiable if it admits at least one "finished" component. In contrast, we show that *termination* is decidable in polynomial time. Termination is a stronger property than weak satisfiability; a terminating complex is weakly satisfiable, but a complex may be weakly satisfiable without terminating. (Jonoska et al. also consider stronger notions of satisfiability, but these do not relate to termination.)

3 The sticker-complex data model

From the outset we assume a finite alphabet Σ. As customary in formal models of DNA computing [21], each letter represents a *domain*, i.e., a string over the DNA alphabet $\{A, C, G, T\}$. The set of resulting domains must form a set of DNA codewords [11, 27, 31]. This should always be kept in mind.

The alphabet Σ is matched with its negative version $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$, disjoint from Σ. Thus there is a bijection between Σ and $\bar{\Sigma}$, which is called *complementarity* and is denoted by overlining; we also set $\bar{\bar{a}} = a$ so complementarity is symmetric. Obviously, $\bar{a}$ stands for the Watson-Crick complement of the DNA sequence represented by a. The elements of Σ are called *positive symbols* and the elements of $\bar{\Sigma}$ are called *negative symbols.*

We recall some fundamental definitions from our previous paper [16], suitably simplified according to the focus of the present paper. The simplifications are only for the purpose of presentation, and our results can be adapted to the original data model, which provides facilities for immobilizing and blocking specific pieces of a complex.

The overall structure of a DNA complex is abstracted in the notion of *pre-complex*. Formally, a pre-complex is a 4-tuple (V, L, λ, μ) where

1. V is a finite set of nodes;
2. $L \subseteq V \times V$ is a finite set of directed edges without self-loops (i.e., (v, v) is not in L for all $v \in V$);
3. $\lambda : V \to \Sigma \cup \bar{\Sigma}$ is a total function labeling the nodes;
4. $\mu \subseteq \{\{v, w\} \mid v, w \in V \text{ and } v \neq w\}$ is a partial matching on the nodes, i.e., each node occurs in at most one pair in μ. Note that the pairs in μ are unordered.

Let C be a pre-complex as above. A *strand* of C is simply a connected component of the directed graph (V, L), so ignoring μ. The *length* of a strand is its number of nodes. A *sticker complex* now is a pre-complex satisfying the following restrictions:

1. Each node has at most one incoming and at most one outgoing edge. Thus, each strand has the form of a chain or a cycle.
2. Strands are homogeneously labeled, in the sense that either all nodes are labeled with positive symbols, or all with negative symbols. Naturally, a strand with positive (negative) symbols is called a positive (negative) strand.
3. Every negative strand has length one or two; if it has length two, then it must have a single edge (i.e., it cannot be a 2-cycle). Negative strands are also referred to as "stickers".
4. Matchings by μ only occur between complementarily labeled nodes: formally, if $\{x, y\} \in \mu$ then $\lambda(y) = \overline{\lambda(x)}$.

In this way, the edges of a sticker complex indicate the sequence order within strands, and the matching μ makes explicit where stickers have annealed to positive strands.

We will also refer to sticker complexes simply as "complexes".

Example 1. A simple example of a complex is depicted in Fig. 1. The alphabet used is $\{a, b, c\}$ with $\bar{a}$, $\bar{b}$ and $\bar{c}$ indicated in the figure as A, B and C, respectively. We will use this convention of showing complementary symbols by capitalizing the symbols, throughout the figures in this paper. The complex consists of ten nodes $x_1, \ldots, x_{10}$, labeled as follows:

node x :	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
label $\lambda(x)$:	a	b	$\bar{a}$	$\bar{b}$	$\bar{c}$	a	b	c	a	a

The nodes are organized in five strands: the negative strand $\bar{a}$ of length 1, two copies of the positive strand ab of length 2; the negative strand $\bar{b}\bar{c}$ of length 2; and the positive strand caa of length 3. More formally, we have

$$L = \{(x_1, x_2), (x_4, x_5), (x_6, x_7), (x_8, x_9), (x_9, x_{10})\}.$$

The matching μ contains the two unordered pairs $\{x_2, x_4\}$ and $\{x_5, x_8\}$.

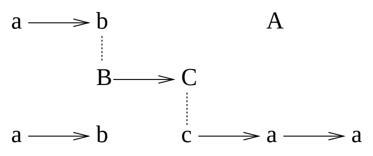

Fig. 1. Example of a sticker complex. Capitalized letters A, B, and C denote complemented symbols a, b, and c. The dotted lines denote the matching μ.

Remark 1. Because stickers are short, there is no need in our model to require that annealed stickers run in complementary ($5'$–$3'$ vs $3'$–$5'$) directions with respect to the positive strands they are annealed to. Indeed, for a sticker of length one, the complementarity is already built into the label; stickers of length two can fold so as to run in complementary direction. Fig. 2 gives an illustration. □

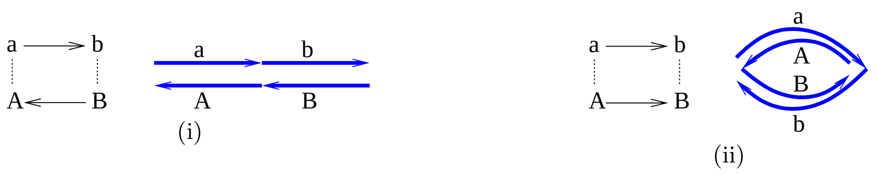

Fig. 2. On the left of (i) and (ii), two different complexes; on the right of (i) and (ii), a depiction of their respective plausible realizations in DNA (recall that each node in the complex represents a DNA sequence, depicted here as thick blue lines).

Note that, in a complex, not all nodes that can be matched must be matched: for example, in Fig. 1, the sticker $\bar{a}$ is not annealed, but could anneal to the four different nodes labeled a. Indeed, it is the hybridization operation, defined below, that will perform all possible matchings.

Components and redundancy. We say that two strands s and s' in a complex are *bonded* if there exists some node v in s and some node v' in s' with $\{v, v'\} \in \mu$. When two strands are connected, possibly indirectly, by this bonding relation, we say they belong to the same component. Thus, a *component* of a pre-complex is a substructure formed by a maximal set of strands connected by the bonding relation. Put another way, whereas a *strand* was defined as a connected component ignoring μ, a component is a connected component *not* ignoring μ.

Example 2. The complex from Example 1 has three components: one consisting of the single strand $\bar{a}$, one consisting of the single strand ab, and one formed by the three strands ab, $\bar{b}\bar{c}$ and caa. □

The intention of our model is that a complex defines the structural content of a test tube. The test tube, however, will in practice hold copies in surplus quantity of each component. Thus, each component of a complex stands for possibly multiple occurrences. We formalize this intention using the notions of subsumption, equivalence, and minimality.

A complex C is said to *subsume* a complex C' if for each component D' of C', there exists an component D in C that is isomorphic to D'. Two complexes C and C' are said to be *equivalent* if they subsume each other. A component D of a complex C is called *redundant* if some other component of C is isomorphic to D. Note that removing a redundant component from C yields a complex that is still equivalent to C.

Remark 2. Isomorphism of sticker complexes can be decided in polynomial time by depth-first search. Indeed, if C and C' both consist of a single component, v is a node of C, and v' is a node of C', then there is at most one isomorphism from C to C' mapping v to v', and this isomorphism can be traced out by depth-first search, following the chain or cycle shape of strands, and the partial matching μ. Depth-first search is in linear time, which yields an isomorphism check for single components in cubic time (try all combinations of v and v'). This algorithm then easily extends to complexes C and C' with multiple components, by matching the components of C to the components of C'. This efficient isomorphism check is in contrast to the problem of general graph isomorphism, which is not known to be decidable in polynomial time. We thus see that sticker complexes form a restricted family of graphs.

4 Hybridization

We give a purely declarative definition of hybridization, in a few steps. We define the two auxiliary notions of "hybridization extension" and "redundant variation". This will allow us to define the fundamental notion of "multiplying hybridization extension (MHE)". The final results of hybridization are then defined as the "saturated" MHEs; those that consist only of "finished" components.

Let $C = (V, L, \lambda, \mu)$ and $C' = (V', L', \lambda', \mu')$ be two complexes. We call C' a *hybridization extension* of C if $V' = V$, $L' = L$, $\lambda' = \lambda$, and μ' is an extension of μ, i.e., $\mu' \supseteq \mu$. A complex C' is said to have *maximal matching* if the only hybridization extension of C' is C' itself.

Example 3. The complex from Example 1 does not have maximal matching; we can properly extend it by adding the pair $\{x_3, x_9\}$ to μ. Alternatively, instead of x_9, we could have taken x_1, or x_6, or x_{10}. Thus the complex has, apart from itself (which is a trivial hybridization extension), four different (non-equivalent) hybridization extensions. These four extensions all have maximal matching, since x_3 is the only negatively labeled node that is not yet matched. $\square$

Let C and C' again be complexes. We call C' a *redundant variation* of C, simply if C subsumes C'. Note that C' may contain redundant components.

Hence, the recipe to produce a redundant variation is simply to take, for every component of C, zero, one, or more copies.

Hybridization is now defined in terms of *multiplying hybridization extensions (MHEs)*, which, by applying redundant variations, account for the presence of surplus copies of components participating in the hybridization. Let C and C' again be two complexes. We call C' an MHE of C if C' is a hybridization extension of some redundant variation C'' of C.

The notion of MHEs is invariant under equivalence, both on the input side as on the output side:

Proposition 1. *Let C_1 and C_2 be two equivalent complexes.*

1. *A complex C' is an MHE of C_1 if and only if C' is an MHE of C_2.*
2. *C_1 is an MHE of a complex C if and only if C_2 is an MHE of C.*

We are not quite finished with the notion of MHE, however. Indeed, an MHE may have "unfinished" components. Formally, we call a component D of an MHE *unfinished* if there exists another MHE in which D occurs bonded within a larger component; otherwise it is called *finished*. An MHE without any unfinished components is called *saturated*.

Example 4. None of the four hybridization extensions of the complex discussed in Example 3 is saturated. Indeed, as long as a component has an unmatched a, that component is unfinished because of we can add a copy of the sticker $\bar{a}$. Specifically, we can finish the large component (consisting of the strands ab, $\bar{b}\bar{c}$, and caa) by matching each unmatched a to a fresh copy of $\bar{a}$, yielding the finished MHE component shown in Fig. 3 (left). Likewise we can finish the component consisting of the single strand ab by matching the a to a copy of $\bar{a}$, as shown in Fig. 3 (right). Finishing the component consisting of the single sticker $\bar{a}$ can be done in two ways: by bringing in a copy of the large component, we get the same result as finishing that large component, and by bringing in a copy of the strand ab, we get the same result as finishing that strand. We conclude that there are precisely two distinct finished MHE components.

Example 5. A complex may have a large number of different finished MHE components: exponentially many in the size of the complex. For example, consider the complex C_n consisting of the following strands:

- a positive strand $a \dots a$ of length n consisting of n nodes all labeled a;
- a sticker $\bar{a}\bar{b}$;
- a sticker $\bar{a}\bar{c}$.

Up to equivalence, there are precisely 2^n finished MHE components for C_n. Each possibility is obtained by annealing, to each node of the positive strand, a copy of either the first or the second sticker. □

We finally define:

Definition 1. *Let C be a complex. The* hybridization *of C equals the disjoint union of all finished MHE components for C.*

$$\begin{tikzcd}
a \arrow[r] \arrow[d, dash, dotted] & b \arrow[d, dash, dotted] & & & & & A \arrow[d, dash, dotted] & \\
A & B \arrow[r] & C \arrow[d, dash, dotted] & A \arrow[d, dash, dotted] & A \arrow[d, dash, dotted] & & a \arrow[r] & b \\
 & & c \arrow[r] & a \arrow[r] & a & & &
\end{tikzcd}$$

Fig. 3. Finished MHE components for the complex shown in Fig. 1.

Termination. A fundamental issue regarding the above definition is that the result of hybridization as defined may be infinite, as shown next.

Example 6. Consider the simple complex consisting of two strands ab and $\bar{b}\bar{a}$ and no matchings. For any number n, using n copies of ab and n copies of $\bar{b}\bar{a}$, we can produce the MHE component shown in Fig. 4 for $n = 3$. This component could also be finished, by matching the remaining a shown on the left with the remaining $\bar{a}$ on the right, effectively creating a ring structure. (As always, in the figure, $\bar{a}$ and $\bar{b}$ are shown as A and B.) Different numbers n yield nonequivalent (non-isomorphic) MHE components, thus the number of potential MHE components is infinite. □

$$\begin{tikzcd}
a \arrow[r] & b \arrow[d, dash, dotted] & a \arrow[r] \arrow[d, dash, dotted] & b \arrow[d, dash, dotted] & a \arrow[r] \arrow[d, dash, dotted] & b \arrow[d, dash, dotted] & \\
 & B & A \arrow[l] & B & A \arrow[l] & B & A \arrow[l]
\end{tikzcd}$$

Fig. 4. Illustration for Example 6.

Nature will compute the result of hybridization by composing MHE's using the available material in the test tube. When, for a given complex C, there are actually infinitely many nonequivalent MHE's, we say that *hybridization does not terminate for C*, or shorter, that *C is nonterminating*; otherwise, we say that *hybridization terminates*, or shorter, that *C is terminating.*

Example 7. So, the complex discussed in the previous example is nonterminating. In contrast, the example complex of Fig. 1 is terminating, as we have seen in Example 4. Also the complexes C_n discussed in Example 5 are terminating. □

In practice, when we have termination of hybridization, a test tube prepared with sufficient quantities of each component of the complex holds, in principle, sufficient material to produce all molecular species that can be the result of hybridization. If sufficient quantities are present, adding even more material will not yield new results. Of course, in practice, a test tube is always finite and the hybridization reaction will, under normal conditions, always "terminate" (reach equilibrium). But the point is that, when hybridization does not terminate for a complex, adding ever more material can, in principle, result in ever more new molecular species (MHE components) to be produced. In this sense, the potential result of the hybridization is indeed infinite.

5 Deciding termination

When designing DNA complexes for DNA computing, it is of course highly desirable to recognize easily whether or not a given complex is terminating. Our main result is the following.

Theorem 1. *A complex is terminating if and only if its hybridization graph does not contain an alternating cycle.*

Corollary 1. *Termination of hybridization is decidable in polynomial time.*

We still need to define the relevant terms used in our theorem, i.e., "hybridization graph" and "alternating cycle". The Corollary will follow since the hybridization graph has the same number of nodes as the given complex, and checking for the presence of an alternating cycle can be done in polynomial time.

The hybridization graph of a complex is an instance of a "partitioned graph". A *partitioned graph* in general is a triple (V, π, E) where (V, E) is an undirected graph and π is a partition of the node set V. Recall that an undirected graph (V, E) consists of a set V of nodes and a set $E \subseteq \{\{v, w\} \mid v, w \in V \text{ and } v \neq w\}$ of unordered pairs of nodes (undirected edges). Recall that a partition of a set V is a set of nonempty, pairwise disjoint subsets of V, called *blocks*, such that their union equals V.

Now given a complex C, the *hybridization graph for* C is the partitioned graph $H = (V, \pi, E)$ defined as follows:

- V equals the set of nodes of C;
- π contains, for each component D of C, the set of nodes belonging to D as a block;
- Let $F \subseteq V$ be the set of "free" nodes of C; a node is called *free* if it is not matched to another node by μ. Then E equals $\{\{v, w\} \mid v, w \in F \text{ and } \lambda(w) = \overline{\lambda(v)}\}$.

Thus, whereas the matching μ in C represents the pairs of nodes that are *already* annealed, the set E contains the pairs of nodes that *may* still be annealed (typically, in an MHE of C).

Example 8. The hybridization graph for the complex of Fig. 1 is shown in Fig. 5. The blocks are depicted as hyperedges (closed curves enclosing the nodes belonging to the same block). The undirected edges are shown as dashed lines. □

The notion of alternating cycle can be defined in general in any partitioned graph $G = (V, \pi, E)$. A *path* in G is a sequence of nodes $v_1, \dots, v_n$ such that for each i with $1 \leq i < n$, we have either an

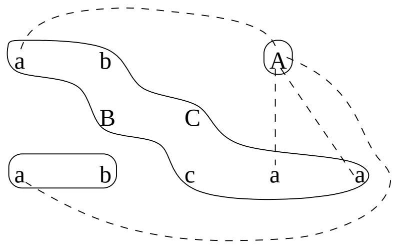

Fig. 5. Example of a hybridization graph.

edge move: $\{v_i, v_{i+1}\} \in E$, or a
block move: $v_i \neq v_{i+1}$ and they belong to a common block.

The path is said to be *alternating* if edge moves happen for each odd i, and block moves happen for each even i (always for $1 \leq i < n$). When the path is alternating, it is said to be an *alternating cycle* when n is odd and at least 3, and $v_n = v_1$.

Example 9. Consider the hybridization graph for the complex of Fig. 1, as shown in Fig. 5. We refer to the node identifiers given in Example 1. Two examples of alternating paths are $p_1 = x_3, x_9, x_1, x_3$ and $p_2 = x_3, x_1, x_{10}, x_3, x_6, x_7$. Note that p_1 is not an alternating cycle; although it satisfies $v_n = v_1$, its length, 4, is not odd. Indeed, this hybridization graph does not admit an alternating cycle, since the only free node with a negative label, $\bar{a}$, is in a component by itself.

Example 10. Consider the complex discussed in Example 6. Its hybridization graph has four nodes partitioned in two blocks. One block, corresponding to the component ab, consists of two nodes x_1 and x_2 labeled a and b, respectively; the second block, corresponding to the component $\bar{b}\bar{a}$, consists of two nodes y_1 and y_2 labeled $\bar{b}$ and $\bar{a}$, respectively. There are two undirected edges, namely, $\{x_1, y_2\}$ and $\{x_2, y_1\}$. This hybridization graph admits an alternating cycle in the form of x_1, y_2, y_1, x_2, x_1. □

The above two examples are in line with Theorem 1. Indeed, the complex of Fig. 1 is terminating, and indeed its hybridization graph does not have an alternating cycle; the complex of Example 6 is nonterminating, and indeed its hybridization graph has an alternating cycle.

The only-if implication of Theorem 1 is relatively easy to prove. The proof of the if-implication (omitted) involves a constructive characterization of MHE components in the form of "hybridization templates", which we present here.

We first need the following auxiliary notion. Let $G = (V, E)$ and $G' = (V', E')$ be two undirected graphs, and let $f : V \to V'$ be a mapping. Then f is called a *semi-strong homomorphism from G to G'* if, for all $u, v \in V$, we have the following:

– if $\{u, v\} \in E$ then $\{f(u), f(v)\} \in E'$; and

– if $\{f(u), f(v)\} \in E'$ then $\{u, w\} \in E$ for some $w \in V$, or $\{v, w\} \in E$ for some $w \in V$.

The first condition is the standard requirement for homomorphisms; the converse of that condition would state the standard requirement for what is known in universal algebra as a "strong" homomorphism. The second condition, however, states only a weak converse (hence the name "semi-strong"), in the sense that if there is an edge between $f(u)$ and $f(v)$, then either u or v have to be involved in an edge, but not necessarily with each other.

Now let $C = (V, L, \lambda, \mu)$ be a complex with hybridization graph $H = (V, \pi, E)$. A *hybridization template for* C is a pair $T = (t, f)$ where $t = (V^t, \pi^t, E^t)$ is a partitioned graph and f is a semi-strong homomorphism from (V^t, E^t) to (V, E), such that:

1. t is connected, i.e., there is a path between any two distinct nodes (using the notion of path in partitioned graphs as defined earlier);
2. E^t is a partial matching, i.e., each node of V^t occurs in at most one edge in E^t; and
3. for each block q of π^t there is a block q' of π such that the restriction $f|_q$ of f to q is a bijection from q to q', i.e., $f|_q$ is injective and the image of $f|_q$ equals q'.

From a hybridization template $T = (t, f)$ for C, and C itself, we can construct a sticker complex $comp(T) = (V^T, L^T, \lambda^T, \mu^T)$ as follows:

– $V^T = V^t$;
– $L^T = \{(x, y) \mid x \text{ and } y \text{ belong to a common block and } (f(x), f(y)) \in L\}$;
– $\lambda^T(x) = \lambda(f(x))$;
– $\mu^T = E^t \cup \{\{x, y\} \mid x \text{ and } y \text{ belong to a common block and } \{f(x), f(y)\} \in \mu\}$.

Proposition 2. *The MHE components are exactly the complexes of the form* $comp(T)$ *with* T *a hybridization template.*

The proof of Theorem 1 also invokes the following lemma which may be interesting in its own right:

Lemma 1. *Let* H *be a partitioned graph with* c *distinct blocks. If* H *admits no alternating cycle, then the length of any alternating path in* H *is at most* $4c + 2$.

6 Complexity issues

Assume hybridization terminates for a given sticker complex C. Then two follow-up questions come up related to the complexity of the result of hybridization. How many finished MHE components can there be? And, how large can a single finished MHE component become?

As we have already seen in Example 5, the *number* of finished MHE components may well grow exponentially in the size of the complex. Also the *size* of MHE components can grow exponentially (details omitted). Unlike Example 5, however, the latter can only happen when the alphabet is allowed to grow with the size of the complex. Usually, however, the alphabet is fixed by the application setting. Indeed we show: (proof omitted)

Proposition 3. *Over the class of terminating complexes over any fixed alphabet, the size of the largest MHE component for a complex C grows only polynomially in the size of C.*

Interestingly, the proof of this proposition relies on the following counterpart to Lemma 1. The two lemmas are complementary as Lemma 1 does not assume anything about the alphabet, whereas Lemma 2 does not assume anything about the complex.

Lemma 2. *Let H be the hybridization graph of a complex over positive alphabet Σ. Let s be the number of symbols in Σ. If H admits no alternating cycle, then the length of any alternating path in H is at most $8s + 2$.*

Remark 3. Since the number of possible graphs on a polynomial number of nodes is singly-exponential, as a corollary to Proposition 3, we obtain that over the class of terminating complexes over a fixed alphabet, the number of MHE components for a complex C is bounded from above by $2^{n^{O(1)}}$, where n is the size of C. Hence, Example 5 essentially illustrates the worst that can happen, i.e., double-exponential or worse is impossible. □

Our final result presents a restriction on classes of complexes, which we call "c-bounded choice" (for a natural number c), so that hybridization is polynomial on the class of c-bounded complexes. It remains to be investigated further how practicable this restriction is, i.e., how many applications can be modeled using sticker complexes that are c-bounded for some c. A positive indication is that only 4-bounded complexes are needed to simulate the relational algebra; to verify this we have inspected the procedures given in an earlier paper [16].

To define the notion of c-boundedness, we first need the notion of a "choice node" of a complex. This is a free node having at least two neighbors in the hybridization graph. Since the edges of the hybridization graph are solely defined in terms of free nodes and their labels being complementary, we see the following, for any label $a \in \Sigma \cup \bar{\Sigma}$: a node v labeled a is a choice node if and only if it is free and there exist at least two free nodes labeled $\bar{a}$. Consequently, if there are at least two free nodes labeled $\bar{a}$, then *all* free nodes labeled a are choice nodes; in the other case, *no* node labeled a is a choice node.

Now for any natural number c, we say that a complex C *has c-bounded choice*, or shorter, *is c-bounded*, if for each component D of C, the number of choice nodes reachable by alternating paths from any node in D, is at most c. Here, naturally, we say that a node w is reachable by an alternating path from a node v, if there is an alternating path starting with v and ending with w. In particular, any node is reachable from itself by an alternating path, since the length-one path v is a trivial alternating path.

Example 11. Recall the complexes C_n discussed in Example 5. Recall that the number of finished MHE components for C_n is 2^n. Since there are two free $\bar{a}$-nodes, the n nodes labeled a are all choice nodes. As these n nodes all belong to a common component, the smallest c such that C_n is c-bounded is n. Hence, there is no fixed c such that all C_n, for all n, are c-bounded.

Suppose now, we modify C_n to C'_n by removing the sticker $\bar{a}\bar{c}$. Then the a-nodes are no longer choice nodes. The only remaining choice node C'_n is the $\bar{a}$-labeled node. Hence, each C'_n is 1-bounded. Now note that each C'_n has only one finished component, obtained by annealing each a-node to the $\bar{a}$-node of a fresh copy of the sticker $\bar{a}\bar{b}$. In particular, hybridization is not exponential on the class of C'_n complexes for all n. □

The above example illustrates our result: (proof omitted)

Theorem 2. *Let c be a natural number. Over the class of terminating, c-bounded complexes over a fixed alphabet, the hybridization of any complex C has size polynomial in the size of C.*

Remark 4. Theorem 2 states that for c-bounded terminating complexes over a fixed alphabet, the result of hybridization has polynomial size. By Definition 1 and Proposition 3, this is the same as saying that the number of finished MHE components is polynomial. Note that it is *not* true that the number of *unfinished* MHE components is polynomial. For example, for each number n, consider a complex U_n with two components: one is the strand $a \dots a$ (n times), and the other is the sticker $\bar{a}$. There are $2^n - 1$ unfinished MHE components, by choosing a strict subset of the n positive nodes, and annealing to each of them a copy of the sticker. There is, however, a unique finished MHE component, obtaining by annealing a copy of the sticker to *all* positive nodes.

Remark 5. There is no converse to Theorem 2 in the sense that, if the result of hybridization has polynomial size over some class K of complexes over some fixed alphabet, then the complexes in K must be c-bounded for some fixed c. Take, for example, the class K consisting of all complexes L_n, for every number n, where L_n consists of four components: a strand $d \dots d$ of length 2^n; a strand $a \dots a$ of length n; and two stickers $\bar{a}\bar{b}$ and $\bar{a}\bar{c}$. The size of L_n is $2^n + n + 4$, and there are 2^n finished MHE components for L_n, which is a number polynomial in the size of L_n. Yet, the class K is not c-bounded for any fixed c, since L_n contains n choice nodes.

7 Conclusion

A natural extension of our approach would be to account for probabilities or error rates on the results produced (finished or unfinished) during hybrization. Of course, error modeling in DNA computation, and secondary structure prediction, are well-known research problems, e.g., [13, 9].

In previous work [16] two of us have defined a database-oriented DNA programming language, called DNAQL, with the goal of understanding the database side of DNA computing. Various open problems remain in connection with this language, including guaranteeing well-definedness through a type system, and understanding the expressive power.

Obviously, we would also like to see the sticker complex data model justified physically (or understand what are the unrealistic aspects), either experimentally or by simulation.

Acknowledgment We thank the program committee for referring us to the work of Jonoska et al. [19].

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science 226, 1021–1024 (Nov 1994)
3. Amos, M.: Theoretical and Experimental DNA Computation. Springer (2005)
4. Arita, M., Hagiya, M., Suyama, A.: Joining and rotating data with molecules. In: Proceedings 1997 IEEE International Conference on Evolutionary Computation. pp. 243–248
5. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E.: An autonomous molecular computer for logical control of gene expression. Nature 429, 423–429 (2004)
6. Boneh, D., Dunworth, C., Lipton, R., Sgall, J.: On the computational power of DNA. Discrete Applied Mathematics 71, 79–94 (1996)
7. Cardelli, L.: Abstract machines in systems biology. In: Transactions on Computational Systems Biology III, Lecture Notes in Computer Science, vol. 3737, pp. 145–178. Springer (2005)
8. Cardelli, L.: Strand algebras for DNA computing. In: Deaton and Suyama [12], pp. 12–24
9. Chen, H.L., Kao, M.Y.: Optimizing tile concentrations to minimize errors and time for DNA tile self-assembly systems. In: Sakakibara and Mi [28], pp. 13–24
10. Chen, J., Deaton, R., Wang, Y.Z.: A DNA-based memory with in vitro learning and associative recall. Natural Computing 4(2), 83–101 (2005)
11. Condon, A., Corn, R., Marathe, A.: On combinatorial DNA word design. Journal of Computational Biology 8(3), 201–220 (2001)
12. Deaton, R., Suyama, A. (eds.): Proceedings 15th International Meeting on DNA Computing and Molecular Programming, Lecture Notes in Computer Science, vol. 5877. Springer (2009)
13. Dimitrov, R., Zuker, M.: Prediction of hybridization and melting for double-stranded nucleic acids. Biophysical Journal 87, 215–226 (2004)
14. Dirks, R., Pierce, N.: Triggered amplification by hybridization chain reaction. Proceedings of the National Academy of Sciences 101(43), 15275–15278 (2004)
15. Garcia-Molina, H., Ullman, J., Widom, J.: Database Systems: The Complete Book. Prentice Hall (2009)
16. Gillis, J., Van den Bussche, J.: A formal model of databases in DNA. In: Horimoto, K., Nakatsui, M., Popov, N. (eds.) Algebraic and Numeric Biology 2010. Lecture Notes in Computer Science, Springer (2011), to appear; for a preprint see http://alpha.uhasselt.be/~vdbuss/dnaql.pdf

17. Hartmanis, J.: On the weight of computations. Bulletin of the EATCS 55, 136–138 (1995)
18. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (1979)
19. Jonoska, N., McColm, G., Staninska, A.: On stoichiometry for the assembly of flexible tile DNA complexes. Natural Computing (2010), published online, 23 January
20. Majumder, U., Reif, J.: Design of a biomolecular device that executes process algebra. In: Deaton and Suyama [12], pp. 97–105
21. Paun, G., Rozenberg, G., Salomaa, A.: DNA Computing. Springer (1998)
22. Qian, L., Soloveichik, D., Winfree, E.: Efficient Turing-universal computation with DNA polymers. In: Sakakibara and Mi [28], pp. 123–140
23. Reif, J.: Parallel biomolecular computation: models and simulations. Algorithmica 25(2–3), 142–175 (1999)
24. Reif, J., et al.: Experimental construction of very large scale DNA databases with associative search capability. In: Jonoska, N., Seeman, N. (eds.) Proceedings 7th International Meeting on DNA Computing. Lecture Notes in Computer Science, vol. 2340, pp. 231–247. Springer (2002)
25. Rothemund, P.: A DNA and restriction enzyme implementation of Turing machines. In: Lipton, R., Baum, E. (eds.) DNA Based Computers: DIMACS Workshop, held April 4, 1995. pp. 75–120. American Mathematical Society (1996)
26. Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N., Goodman, M., Rothemund, P., Adleman, L.: A sticker-based model for DNA computation. Journal of Computational Biology 5(4), 615–629 (1998)
27. Sager, J., Stefanovic, D.: Designing nucleotide sequences for computation: A survey of constraints. In: Carbone, A., Pierce, N. (eds.) Proceedings 11th International Meeting on DNA Computing. Lecture Notes in Computer Science, vol. 3892, pp. 275–289. Springer (2006)
28. Sakakibara, Y., Mi, Y. (eds.): Proceedings 16th International Conference on DNA Computing and Molecular Programming, Lecture Notes in Computer Science, vol. 6518. Springer (2011)
29. Sakamoto, K., et al.: State transitions by molecules. Biosystems 52, 81–91 (1999)
30. Seelig, G., Soloveichik, D., Zhang, D., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 315(5805), 1585–1588 (2006)
31. Shortreed, M., et al.: A thermodynamic approach to designing structure-free combinatorial DNA word sets. Nucleic Acids Research 33(15), 4965–4977 (2005)
32. Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. PNAS (2010), published online, 4 March
33. Soloveichik, D., Winfree, E.: The computational power of Benenson automata. Theor. Comput. Sci. 244(2–3), 279–297 (2005)
34. Winfree, E., Yang, X., Seeman, N.: Universal computation via self-assembly of DNA: Some theory and experiments. In: Landweber, L., Baum, E. (eds.) DNA Based Computers II: DIMACS Workshop, held June 10–12, 1996. pp. 191–213. American Mathematical Society (1998)
35. Yamamoto, M., et al.: Development of DNA relational databases and data manipulation experiments. In: Mao, C., Yokomori, T. (eds.) Proceedings 12th International Meeting on DNA Computing. Lecture Notes in Computer Science, vol. 4287, pp. 418–427. Springer (2006)