Solving Generic Decision Problems by in-Message Computation in DNA-Based Molecular Nanonetworks

Florian-Lennert Adrian Lau(✉), Regine Wendt, and Stefan Fischer

Institute of Telematics, Ratzeburger Allee 160, 23562 Lübeck, Germany
{lau,wendt,fischer}@itm.uni-luebeck.de

Abstract. One of the biggest unsolved problems in nanonetwork research is the actual construction of the components required for building such networks. Most existing ideas are limited to partial solutions of construction of nanodevices, computation within them, and communication between them. While many ideas are promising, the problem remains how to combine those various building blocks into operational and efficient nanonetworks.

In this paper we use DNA as *the* basic building block for all components of nanonetworks. The inherent properties of this molecule are used to assemble complex nanostructures. DNA can be utilized to create both nanodevices and a communication mechanism. Properly designed DNA-molecules can even be utilized for computational purposes. In summary, DNA forms the base for an exhaustive nanonetwork concept.

This work specifically presents an approach how to solve arbitrary mathematical problems that can be modeled as boolean formulas using DNA-based nanonetworks by in-message computation. The computation itself is encoded in the assembly process of a message. This avoids often-stated space constraints for computations at the nanoscale, as the medium of transportation is commonly less constrained than the size of nanodevices dictates. This method thereby presents a constructive approach on how to actually create message molecules, rather than only proving the general possibility.

Keywords: Nanonetworks · Tile-based self-assembly · Nanostructures · Molecular communication · Decision problems

1 Introduction

After the concept of nanonetworks and their use in, e.g., medical applications have been adequately described in recent years, research groups worldwide are now looking into the question of how such nanonetworks can actually be constructed. A promising approach is the paradigm of self-assembly: components of nanonetworks such as sensors, actuators and even the messages exchanged

M. M. Alam et al. (Eds.): BODYNETS 2020, LNICST 330, pp. 121–136, 2020.
https://doi.org/10.1007/978-3-030-64991-3_9

between them can grow like crystals [7] instead of being manually engineered by humans – a process hardly imaginable on the nanoscale [4].

Crystal formation is a natural instance of the principle of self-assembly. It is based on tiny and often simple components assembling themselves into larger and more complex structures according to local rules. In order to make this process usable in the context of building nanonetworks, the resulting complex structures should be nanodevices, nanosensors, message molecules, fully functional nanonetworks or even computers [2]. However, it is hardly possible to influence the process. A very fortunate exception are certain building blocks made from DNA. These also follow the paradigm of self-assembly and can be treated like crystals [16].

The basic entities we are looking at are DNA strands. It is possible to create arbitrary DNA strands in the laboratory [13]. One can actually create DNA building blocks out of DNA strands that behave like puzzle pieces [17]. These puzzle pieces of DNA are also called *tiles* or *DNA-tiles*. Tiles can be designed in such a way that they bind with other tiles only in a specific and predetermined way. It has been shown that this characteristic can be used to create almost any structure at the nanoscale [8]. Additionally, tiles can also perform computations through conditional binding processes [9,17]. All these building processes execute – after an initial tile set has been provided – without the need for any further human intervention [7]. Also, all the materials required for this approach can be produced in a laboratory, so that the first rudimentary nanonetworks can really be built and operated in-vitro.

DNA-based nanonetworks, in their most primitive form, are unfortunately not programmable once the devices have self-assembled. Further, implementing conventional algorithms with just a small set of tiles is much less well researched than conventional programming languages. As a consequence, the purpose of a nanonetwork has to be known before its self-assembly, and the tiles have to be created and selected accordingly. While a proof of concept for this approach has already been presented in [7], it is still mostly unknown how nanonetworks for specific mathematical problems can be created, i.e., given a specific problem formulation, how many different tiles are needed and how they look like.

This paper presents a novel approach on how to create the necessary building blocks for specific, but arbitrary problems, namely those that *can be modeled as a boolean formula*. Previous findings that tile-based self-assembly systems are Turing-complete [17] are of mere theoretical interest, as the proof gives no algorithm on how to compute in a feasible manner [14]. This paper bridges the gap by offering a solution for the complexity classes AC^0 and NC^1. AC^0 and NC^1 are classes of problems that describe strongly restricted circuits of limited size, among other [3,15]. Important problems of these classes are basic arithmetic functions like addition and multiplication, as well as logical operations and the computation of thresholds. These form the basic necessary operations to create more complex communication protocols. Looking at possible applications, nanonetworks can be created that react, upon measuring environmental param-

eters, once a predefined threshold has been reached, thus increasing reliability. A logical AND can be used to establish a distributed consensus among nanodevices, as demonstrated in [7].

The remainder of the paper is structured as follows: Sect. 2 presents the necessary definitions to understand the mathematical models and the process of self-assembly in general. Section 3 introduces the definitions and general ideas behind DNA-based nanonetworks. Section 4 suggest an algorithms that creates message molecules for problems that can be modeled as boolean formulas. Section 5 explains a generic nanonetwork architecture. Section 6 summarizes by presenting a list of problems that can be solved by DNA-based nanonetworks via the developed procedure.

2 Preliminaries for Tile-Based Self-Assembly

This section gives a brief introduction and examples of tile-based self-assembly systems from Eric Winfree [17]. The notation used in the preliminaries is based on [5] and has already been presented in [7]. For a detailed definition of self-assembly systems and an overview of the most important results regarding self-assembly systems, please consult [11].

Tiles are the basic components of self-assembly systems. A 2D-tile is a square in $\mathbb{Z}^2$ and a 3D-tile is a cube in $\mathbb{Z}^3$. Figure 1(a)–(d) show examples of two-dimensional and three-dimensional tiles. Models with focus on the mathematical functionality (a, b) and with focus on the biological components (c, d) are presented.

From here on, the dimension is largely omitted. Unless otherwise specified, two-dimensional tiles are implied by the symbol t.

Each side of a tile can have any number of *glues*, with a corresponding *binding strength*. The glues and their strengths are depicted by a number of black boxes on each side. In Fig. 1 all glues are of strength $s = 1$. The glue color is indicated by a label, here N, E, W and S.

Additionally, each tile has a *marker* in the middle. The marker is also represented by a label and can be implemented by a florescence marker biologically. It is used to encode semantic properties of a tile. An example would be the representation of an encoded truth value.

In the DNA-computing research community, tiles are made from DNA [11]. Glues are implemented by open DNA strands with freely selectable base sequences. This enables the implementation of both color, by the use of a specific base sequence, and strength, by the corresponding length of the open strands.

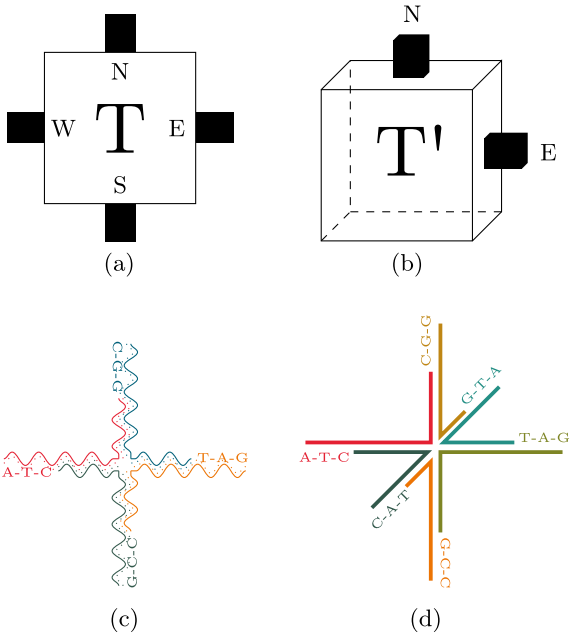

Fig. 1. Examples of tiletypes with mathematical and biological representation in 2D and 3D. The modeled glues are displayed as black cubes, their type as a label. The biological glues are illustrated by base sequences. [1]

Examples of DNA-tiles are shown in Fig. 1(c, d). The open strands with the displayed bases adenine (A), thymin (T), cytosin (C) and guanin (G) can only bind with a matching complementary sequence.

A tile can have no or exactly one neighbor on each side. Tiles t and t' only interact with each other when they are neighbors. The interaction rules are determined by the glues.

If glues are not explicitly shown or mentioned, the empty glue $\mathcal{L}_g = \emptyset$ with the label VOID and the strength 0 is assumed.

A tile t, with a glue strength 0 or no glue at a side can't interact with other tiles t' at that side.

Two tiles are of the same *tiletype*, iff they have identical glue color, glue strength and markers.

The *temperature* τ of a self-assembly system describes the minimum glue strength s that is required for a tile to form a stable bond with other tiles.

Two tiles bind with each other if they are neighbors and have glues of a suitable color and if the binding strength with all neighbors is at least equal to the temperature. Meaning that two tiles can only bind at temperature 2 if they share a glue with the same color and 2 black boxes.

The temperature τ models the physical temperature. If the temperature τ is increased, molecules move faster. The additional energy makes it more likely that bonds are destroyed or cannot form in a stable manner.

In this paper, τ is assumed as 2 or 3 and omitted in the notation. Temperatures 2 and 3 are sufficient to implement all functionality relevant to the scenarios discussed in this paper.

In order for tiles to bind together, they must be rotated to match each other. In biological systems based on DNA this happens automatically. Mathematical models require that tiles are not rotatable.

Due to the intrinsic properties of DNA, errors are unavoidable [17]. However, the number of errors during the self-assembly process can be reduced if special tiletypes are used. In [11], procedures are described that greatly reduce the probability of errors in two dimensions. In [7] this reduction is further explained in the context of nanonetworks and in [1] a procedure for three dimensions is introduced.

The *binding strength* s of tiles t and $t^{'}$ is equal to the number of matching glues with all neighbors of a tile.

The *total binding strength* of a tile t is the sum of binding strengths between t and all of its correct neighbors. Incorrect neighbors do not contribute to the total binding strength. An *assembly* is the result of two or more tiles forming a stable binding with each other.

The *border* of an assembly α is a subset $\alpha^{'}$ of α. The border only contains tiles with at least one unoccupied neighboring position. A tile t correctly binds with an assembly α if the total binding strength of t with neighboring tiles of assembly α is at least equal to the temperature τ. Only border-tiles of assemblies may interact with free tiles.

The *growth front* of n-dimensional assemblies is a subset of positions of $\mathbb{Z}^n$. A position is part of the growth front iff it is unoccupied and neighbor to a border-tile that has a positive glue strength at that side.

The positions of the growth front change when tiles are added to or removed from the assembly. It is assumed that exactly one tile is added or removed at any discrete time.

The initial assembly α_0 at time 0 is called the *seed-assembly* or *seed-tile* σ.

Starting with σ, tiles are added non-deterministically to the assembly at the growth front. Due to the inherent non-determinism of self-assembly systems, it can be difficult to create tilesets T that grow into desired structures.

An *assembly sequence* of a self-assembly system is a sequence $S = \langle \alpha_0, \alpha_1, \ldots \rangle$, while α_{i+1} is created from α_i by adding a tile to α_i. The last element of a finite sequence is called the *result* or *terminal*.

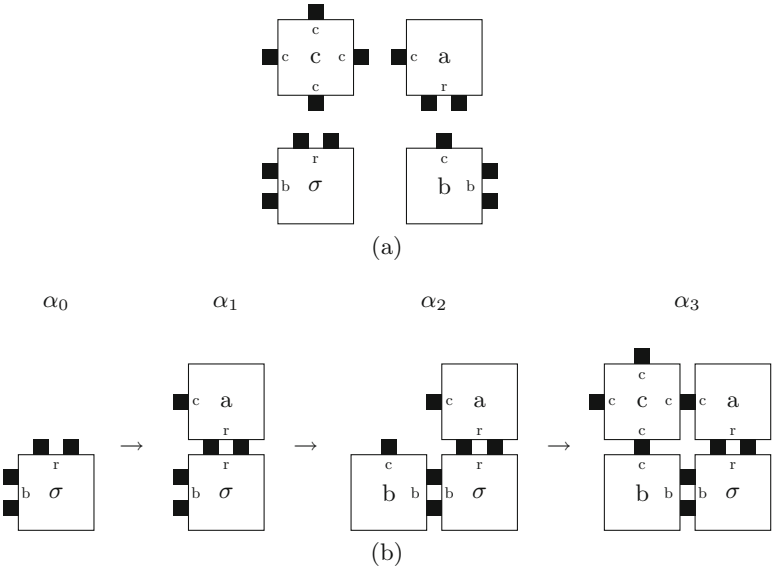

Fig. 2. (a) Example for a tileset of a 2D-self-assembly systems. (b) The corresponding assembly sequence of the tileset from (a). The seed-tile is called σ and the temperature is 2. $S = \langle \alpha_0, \ldots, \alpha_3 \rangle$ shows the three steps until the assembly is terminal. [7]

Figure 2(a) shows an example tileset of a 2D-TAM with seed-tile σ. The assembly sequence of the TAM $\mathcal{T}_\tau$ at temperature $\tau = 2$ is depicted in Fig. 2(b). α_0 describes the assembly at time 0. α_1, α_2 and α_3 show the assembly after the addition of one tile at a time. In α_3, the required correct glues originate from two different neighboring tiles.

3 DNA-Based Nanonetworks

In the following sections different components are defined, which are necessary for the creation of DNA-based nanonetworks. The idea of using DNA-tiles as construction material for nanonetworks, their computations and communication mechanisms goes back to [7]. For a detailed description of DNA-based nanonetworks see [7].

First, a communication mechanism based on tiles is described. This is followed by an explanation of how receptors and ligands can be realized using DNA. The individual components are then combined and formally defined as DNA-based nanonetworks.

3.1 Message Molecules

Since self-assembly systems can also be used as computational models, the presented approach has the advantage that computations can be integrated into the assembly process of a message molecule.

A message molecule can be designed in such a way that specific tiles are necessary to achieve a fully assembled message molecule. By making sure that these tiles are only distributed under certain conditions – e.g. in case of an event – they can be interpreted as input for a computation. Other tiles, also required for the computation, can be kept in the medium as required [7].

This method ensures that, for example, a ligand necessary for a binding reaction is only formed on a message molecule once a computation has been finalized. A ligand is the part of a molecule that can form a bond with a receptor.

Tiles and assemblies are subject to Brownian motion. Brownian motion can be used as a distribution mechanism to transport tiles to the required positions. The process is largely random, which is why a large number of message molecules is required.

Any decision problem [14] can be solved by a self-assembly system at temperatur 2 [17]. However, it is unclear whether nanobots will perform computations in the same way as macroscopic computers and whether there is enough space for complex computations at the nanoscale [6]. Further, it is mostly unknown how to create a space efficient self-assembly system that solves a given decision problem using message molecules. In the following sections a generalized method that creates message molecules for many decision problems is explained.

Since self-assembly systems solve a large number of challenges at once, while they can already be created in the laboratory, it is compelling to use this previously proven technology.

Definition 1. *A message molecule $\mathcal{M}_\Phi$ is a tileset T, which computes the boolean formula Φ and creates a ligand in the case of a successful computation.*

3.2 Forming Ligands

The ligands are binding sites on message molecules, which enable a correct binding with nanobots. These are modeled by the glues of tiles.

The variable size and shape of message molecules require additional tiletypes to always provide a uniform starting point for the composition of a ligand. Furthermore, it should only form if a computation has been successfully completed.

Figure 3 shows an example of a ligand. Since it is a temperature 2 system, the two tiles with the label "R" can only bind to the assembly once the middle row of the adjacent message molecule is complete. This in turn depends on the successful assembly of the remaining assembly [7]. After the whole ligand is finalized, the ligand can bind to the receptor from Fig. 4.

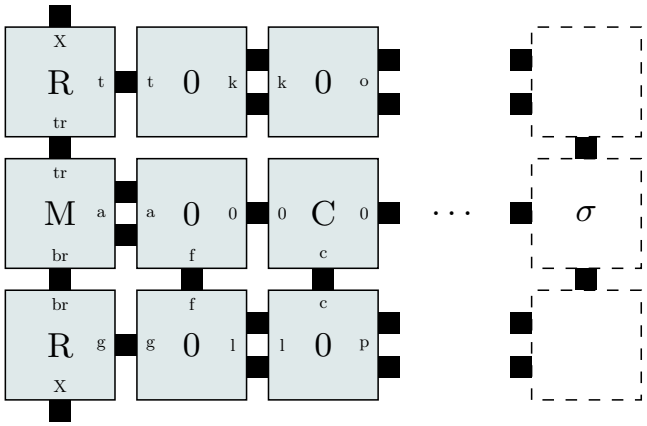

Fig. 3. Generic ligand of a message molecule at temperature 2. The tiles with marker "R" can only attach to the assembly if tile C has a right neighbor. The white tiles represent an adjacent message molecule that has to be finalized before the ligand can form. [7]

3.3 Message Receptors

Receptors are the parts of nanobots that can bind message molecules. They are modeled by tiles with appropriate glues. These can be generated by simple assemblies. Unlike ligands, they are not tied to the successful computation of a formula Φ.

Receptors can be of any shape as long as they bind their corresponding ligand without overlapping. In this case the temperature restriction must be observed. Furthermore, the externally available glues of a receptor must be at least one tile apart from each other to prevent premature binding of parts of the ligand to the receptor.

Figure 4 shows an example of a possible receptor for a message molecule. The gray squares represent any part of a ligand message. The black squares represent individual glues of strength 1. The labels of the receptor's glues encode the binding condition. If the receptor is correctly designed, it can "recognize" the outermost three tiles of the message molecule at once.

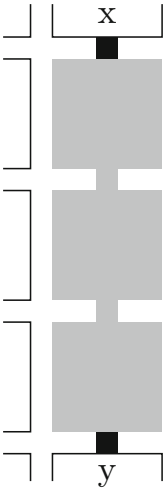

Fig. 4. Generic receptor, which can stably bind message molecules at temperature 2. The grey tiles represent the ligand of a message molecule with unbound glues x and x that are spaced one tile apart from each other. The white tiles are part of a nanobot with unbound glues x and y that allow message molecules to bind. [7]

4 Creating Generic Message Molecules

Although it is known that self-assembly systems are Turing-complete at temperature 2, this finding cannot be transferred directly to message molecules. In this section a procedure is presented, with which many mathematical problems can be compiled to message molecules. This demonstrates the potential versatility of DNA-based nanonetworks.

Theoretical computer science has proven that different types of problems can be transformed into each other. Since decision problems can easily be processed at the nanoscale, these are of particular interest. Theorem 1 proofs a procedure that creates a tileset that assembles into message molecules that solve a given boolean formula Φ.

Theorem 1. *For each decision problem modeled as a boolean formula Φ, a corresponding message molecule can be created that executes the same computation and only forms a ligand if the result of the computation is "1".*

Proof. Let Φ be a boolean formula. We consider every truth assignment that satisfies the formula Φ.

Every boolean formula may be transformed into disjunctive normal form [12]. This is done by setting up the truth table for said formula. Table 1 shows an example.

Table 1. Example truth table for the formula $A \wedge B$.

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

For each line of the truth table that contains "1" in the last column a clause is created. A clause is a formula phi' that only contains literals, negative literals and the logical $\wedge$. The process is repeated for every row of the truth table. All resulting formulas are then combined with the logical $\vee$ operator. The result is a canonical formula that performs an equivalent computation as the original boolean formula Φ. The produced formula is called the disjunctive normal form of a formula Φ. The size of the result can be further reduced by procedures like McCluskey or Karnaugh maps [10].

For every formula in disjunctive normal form a tileset can be generated as follows:

For every sub formula a message molecule is created. The number of rows of each message molecule is equal to the number of literals times 3. Figure 5 shows an example for the formula from Table 1.

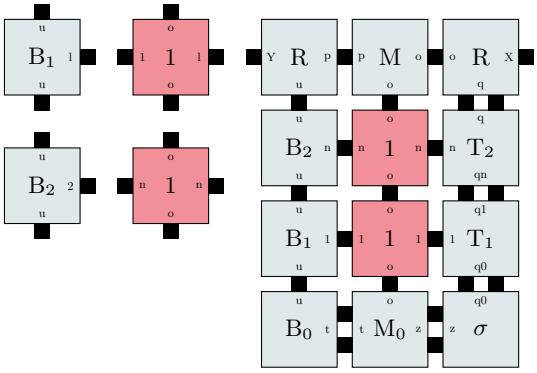

Fig. 5. Example tilesets for the DNF formula from truth Table 1. For every row i of the truth table that evaluates to "1", two tiletypes for each literal are created (left). These assemble into the message molecule (right).

The truth value of each literal is represented by the label of the corresponding tile as well as by its glues. Only if all literals are part of the message molecule a ligand is formed. Each ascending row of the message molecule can only form once the previous row assembled fully.

Since there is a message molecule for each sub formula and all the sub formulas are combined by a logical $\vee$, the completion of a single message molecule is sufficient to communicate a successful computation. □

5 Arbitrary Nanonetworks for Boolean Formulas

In [6] a list of relevant mathematical operations for nanonetworks is presented. Most of them can be solved by representing them as a boolean formula Φ that can be transcribed into a message molecule by applying the methodology from Sect. 4.

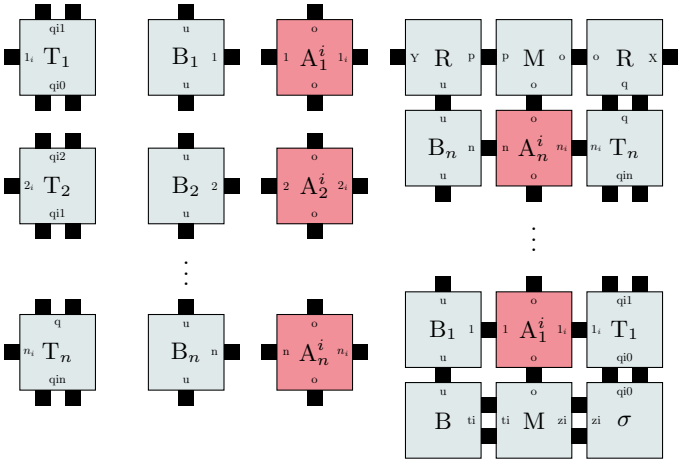

Fig. 6. Tileset (left) and fully assembled message molecules (right) for a boolean formula Φ. The tileset to the left contains three tiletypes for every literal in line i of the corresponding truth table and only contains tiles that are used more than once. To the right, the message molecules for the clauses of the disjunctive normal form (DNF) of the formula Φ are displayed.

Figure 6 shows an example for the general architecture with n input bits. i represents the index of the row of a truth Table 2 that evaluates to "1". The truth table represents an arbitrary boolean formula Φ. The disjunctive normal form of the formula Φ can directly be derived from the table.

The assembly process of the message molecules starts with the seed-tile σ at the bottom right. Since the temperature is 2, The bottom row of M and B can form, as well as the right stack of tiles $T_1 \ldots T_n$. The remaining assembly process can only proceed if the red tiles A_n^i, that represent the literals of each clause, are present. Once all required tiles A_n^i are bound to the assembly the tiles $B_1 \ldots B_n$ can attach, followed by the left tile R, which completes the ligand.

Table 2. Generic truth table for a boolean formula Φ. The columns represent all occurring sub formulas. Each row i that evaluates to "1" represents a truth assignment that satisfies the formula Φ and serves as input for a message molecule. The red line shows an example.

ϕ_1	ϕ_2	...	Φ
1	1		1
1	0		1
		⋮	
0	1		0
0	0		1

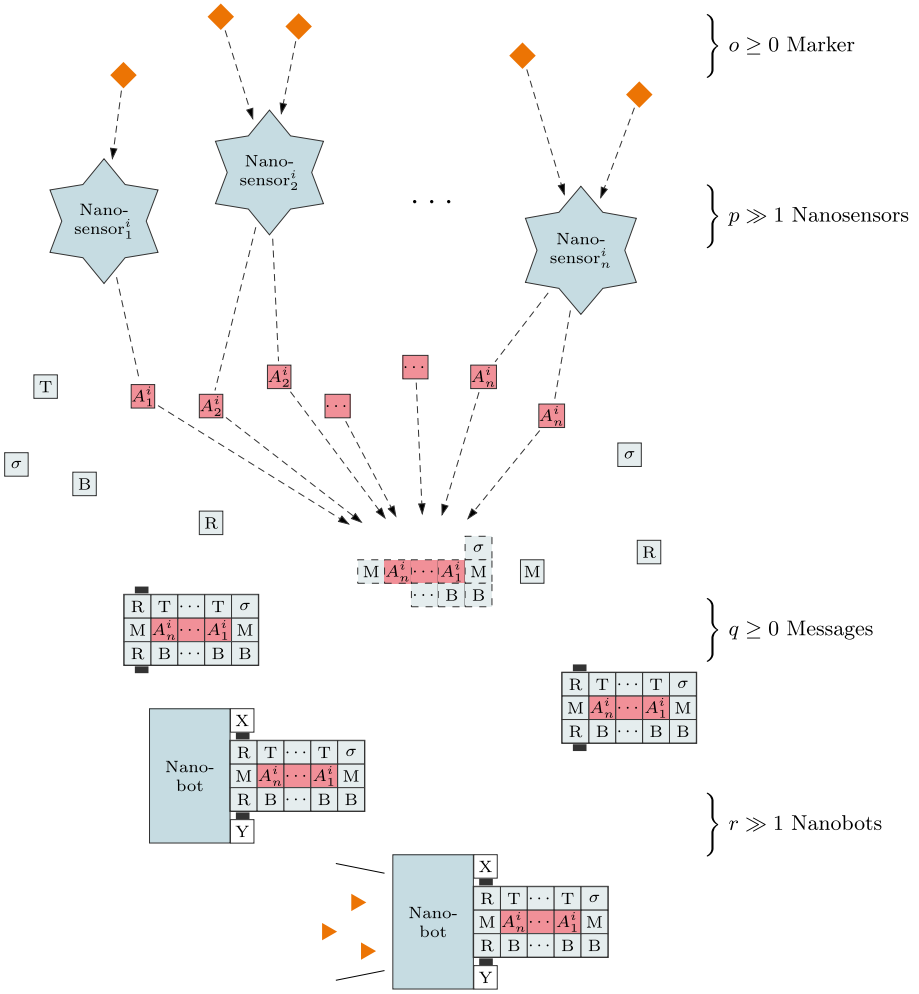

Fig. 7. The general reference architecture for a DNA-based nanonetwork. It consists of $o \geq 0$ markers (orange rhomboids), $p \gg 1$ nanosensors, $q \geq 0$ message molecules and $r \gg 1$ Nanobots. (Color figure online)

The binding of one of the created message molecules to a nanobot suffices to implement the required logical $\vee$. Combined, the result is a behavior that maps the original problem. The corresponding nanonetwork is shown in Fig. 7. It depicts a set of nanosensors$_n^i$ that measure specific markers (orange rhomboids) and release tiles A_n^i into a medium once a successful measurement has been conducted. Together with tiles that are always present in the shared medium, the tiles assemble into message molecules that compute the clauses of the given boolean formula Φ.

Only in the case of a successful computation a ligand is formed and the resulting message molecule can bind to a nanobot which then can release medical payload or communicate the measurement (orange triangles).

Apart from message molecules, receptors and ligands, nanosensors and nanobots can also be created from tiles. Consequently, we define a nanonetwork as a tileset. Nanobots and nanosensors are modeled as hollow cubes, which may be opened once a specific message molecules or markers binding to them.

Definition 2. *A tile-based nanonetwork $\mathcal{N}_\Phi$ is a tileset $\mathcal{N}_{Se} \cup \mathcal{N}_R \cup \mathcal{M}_\Phi$. $\mathcal{N}_{Se}$ is a tileset for nanosensors. $\mathcal{N}_R$ is a tileset for nanobots and $\mathcal{M}_\Phi$ a tileset for message molecules, which compute the function Φ.*

6 Conclusion and Future Work

This work advances the modeling framework for DNA-based nanonetworks presented in [7]. While [7] showed that it is in-fact possible to create nanonetworks that solve arbitrary decision problems, it was not elaborated how to achieve this goal. The viability of the approach was exemplified by the implementation of a DNA-based nanonetwork for the mathematical operation AND with four bits.

In this work we closed the gap by presenting and proving a procedure that creates message molecules for any decision problem that can be modeled by a boolean formula. This increases the number of known, feasible computations by DNA-based nanonetworks tremendously. In [6] a list of mathematical operations that are of interest to nanonetworks has been assembled. Table 3 shows a subset of the entries. The left column shows problems that were derived from medical use-case scenarios, while the right column displays additional problems from the same complexity class. For all of the depicted problems a corresponding decision problem can be defined and solved by our methodology. This can easily be deduced since all of the problems can be solved by circuits, which directly implies that they can be expressed as a boolean formula.

As predicted by [6], the computations of AC^0-messages and NC^1-messages are relatively simple to implement at the nanoscale.

The complexity classes AC^0 and NC^1 describe circuits with specific limitations. Boolean circuits characterized by AC^0 are of polynomial size and constant depth in regard to the number of input bits. The gates of NC^1 have two inputs per gate at most, but can be of logarithmic depth in regard to the number of input bits. For a detailed description consult [6]. Both complexity classes represent comparably easy problems.

Table 3. List of problems sorted by complexity classes. The class NC^1 includes the class AC^0 and the problems in AC^0 are therefore considered *easier*. Both AC^0 and NC^1 describe problems that require very little space to be computed. Table adapted from [6].

	Medical problems	Additional problems
AC^0-message:	ADD	ODD/EVEN
	SUB	DIV_2
	SIGN	MOD_2
	INC	INV
	AND/OR	LOG_2
NC^1-message:	MULT	MIN/MAX
	DIV	PARITY
	EXP	IT-MULT
	MAJOR	MOD
	THRES	
	IT-ADD	
	AVG	

However, the size of a truth table grows exponentially with the number of different literals in the boolean formula that serves as an input. Therefore it isn't always possible to find a small set of message molecules that solves a given problem. However, for many of the presented scenarios our methodology suffices.

Tile-based self-assembly systems are indeed Turing-complete and a universal Turing-machine can be created from tiles. However, the number of required tiletypes for a tileset that encodes a Turing-machine is huge and often infeasible. The presented methodology provides a feasible approach for many common input-sizes.

For mathematical systems, more efficient solutions can often be constructed for specific problems. It is possible and likely that smaller message molecules exist for individual operations. The general approach presented in this paper works for all boolean formulas, specific, fixed operations however, can often be solved in a more efficient manner.

The next logical step is the construction of tilesets for the most important problems from Table 3. The problems ADD, MULT and THRES are of particular interest as they are among the most commonly used operations in modern programming. Space-efficient message molecules for those operations could be combined to encode the behavior of primitive communication protocols.

Furthermore, the feasibility of the presented approach could be tested in a wet-lab experiment. Successfully simulating the assembly process of the message molecules would illustrate the practical usefulness and generality of tile-based nanonetworks, as the presented procedure is capable of solving numerous problems.

References

1. Bende, P., Lau, F., Fischer, S.: Error-resistant scaling of three-dimensional nanoscale shapes on the basis of DNA-tiles. In: Proceedings of the 6th ACM International Conference on Nanoscale Computing and Communication (ACM NanoCom 2019), Dublin, Ireland (2019)
2. Büther, F., Lau, F.-L., Stelzner, M., Ebers, S.: A formal definition for nanorobots and nanonetworks. In: Galinina, O., Andreev, S., Balandin, S., Koucheryavy, Y. (eds.) NEW2AN/ruSMART/NsCC -2017. LNCS, vol. 10531, pp. 214–226. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67380-6_20
3. Clote, P., Kranakis, E.: Boolean Functions and Computation Models. Texts in Theoretical Computer Science. An EATCS Series, 1st edn. Springer, Heidelberg (2002)
4. Kallenbach, N., Ma, R.I., Seeman, N.C.: An immobile nucleic acid junction constructed from oligonucleotides. Nature **305**(5937), 829 (1983)
5. Lathrop, J.I., Lutz, J.H., Summers, S.M.: Strict self-assembly of discrete Sierpinski triangles. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) CiE 2007. LNCS, vol. 4497, pp. 455–464. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73001-9_47
6. Lau, F., Büther, F., Gerlach, B.: Computational requirements for nano-machines: there is limited space at the bottom. In: Proceedings of the 4th ACM International Conference on Nanoscale Computing and Communication, ACM NanoCom 2017, Washington DC, USA (2017)
7. Lau, F., Büther, F., Geyer, R., Fischer, S.: Computation of decision problems within messages in DNA-tile-based molecular nanonetworks. Nano Commun. Netw. **21**, 100245 (2019). https://doi.org/10.1016/j.nancom.2019.05.002
8. Lau, F., Stahl, K., Fischer, S.: Techniques for the generation of arbitrary three-dimensional shapes in tile-based self-assembly systems. Open J. Internet of Things **4**(1), 126–134 (2018). https://www.ronpub.com/ojiot/OJIOT_2018v4i1n10_Lau.html. Special Issue: Proceedings of the International Workshop on Very Large Internet of Things (VLIoT 2018) in Conjunction with the VLDB 2018 Conference in Rio de Janeiro, Brazil
9. Lee, J.Y., Shin, S.Y., Park, T.H., Zhang, B.T.: Solving traveling salesman problems with DNA molecules encoding numerical values. Biosystems **78**(1), 39–47 (2004). https://doi.org/10.1016/j.biosystems.2004.06.00. http://www.sciencedirect.com/science/article/pii/S0303264704001157
10. McCluskey Jr., E.: Minimization of Boolean functions. Bell Syst. Tech. J. **35**(6), 1417–1444 (1956)
11. Patitz, M.J.: An introduction to tile-based self-assembly and a survey of recent results. Natural Computing **13**(2), 195–224 (2013). https://doi.org/10.1007/s11047-013-9379-4
12. Quine, W.: The problem of simplifying truth functions. Am. Math. Monthly **59**(8), 521–531 (1952)
13. Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature **440**(7082), 297–302 (2006). https://doi.org/10.1038/nature04586
14. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proc. London Math. Soc. **2**(1), 230–265 (1937)
15. Vollmer, H.: The NC hierarch. In: Introduction to Circuit Complexity. Texts in Theoretical Computer Science An EATCS Series, pp. 107–171. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-662-03927-4_5

16. Watson, J.D., Crick, F.H.: A structure for deoxyribose nucleic acid. Nature **171**, 737–738 (1953). http://www.nature.com/nature/dna50/watsoncrick.pdf
17. Winfree, E., Liu, F., Wenzler, L., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. Nature **394**(6693), 539–544 (1998)