Computing properties of stable configurations of thermodynamic binding networks*

Keenan Breik[1], Chris Thachuk[2], Marijn Heule[1], David Soloveichik[1]

[1] University of Texas at Austin
[2] California Institute of Technology

Abstract. The promise of chemical computation lies in controlling systems incompatible with traditional electronic micro-controllers, with applications in synthetic biology and nano-scale manufacturing. Computation is typically embedded in kinetics—the specific time evolution of a chemical system. However, if the desired output is not thermodynamically stable, basic physical chemistry dictates that thermodynamic forces will drive the system toward error throughout the computation. The thermodynamic binding network (TBN) model was introduced to formally study how the thermodynamic equilibrium can be made consistent with the desired computation, and it idealizes binding interactions subject to enthalpy and entropy tradeoffs. Here we consider the computational complexity of natural questions about TBNs and develop a practical algorithm for verifying the correctness of constructions by translating the problem into propositional logic and solving the resulting formula. The TBN model together with automated verification tools will help inform strategies for error reduction in molecular computing, including the extensively studied models of strand displacement cascades and algorithmic tile assembly.

1 Introduction

Similar to digital electronics, advances in engineering of molecular computation have relied on a distinctive set of abstractions and models. DNA strand displacement cascades (formalized as [6]) made it possible to rationally design molecular reaction pathways, and this model has been used to engineer a wide range of molecular devices, logic and neural circuits, and dynamical systems [10, 8]. Likewise, the formalism of algorithmic tile assembly [3] enabled the self-assembly of complex 2D and 3D nanostructures [5, 2].

The widely studied models of chemical computing such as strand displacement cascades and algorithmic tile assembly are essentially kinetic as they describe a desired time evolution of an information processing chemical system. However, unlike electronic computation, chemical computation operates in a Brownian environment subject to powerful thermodynamic driving forces. If the desired

* See a draft of the full paper at https://arxiv.org/abs/1709.08731

output happens to be a meta-stable configuration, then thermodynamic driving forces will inexorably drive the system toward error. For example, *leak* in most strand displacement systems occurs because the thermodynamic equilibrium of a strand displacement cascade favors incorrect over the correct output, or does not discriminate between the two [9]. In tile assembly, thermodynamically favored assemblies that are not the intended self-assembly program execution are likewise a major source of error [7, 2].

2 Model

The TBN model abstracts chemical systems at the thermodynamic equilibrium [4]. The model is simple and general due to its two main features: (1) abstracting away of "geometry", and (2) the simplification of thermodynamics to a tradeoff between enthalpy (equated with the number of bonds) and entropy (equated with the number of complexes). If entropy and enthalpy conflict, we assume enthalpy wins. Many systems studied in molecular programming operate in this limit: in particular, DNA systems with long domains at relatively low concentrations [9].

Formally, a TBN is a triple $(D, ^*, T)$. Here D is a finite set we call the *site types*; $^*\colon D \to D$ is an involution, so the *complement* of site type a^* is $(a^*)^* = a$; and T is a finite multiset of monomer types, where a *monomer type* is a finite multiset over D. We often call T alone a TBN when D and * are to be inferred.

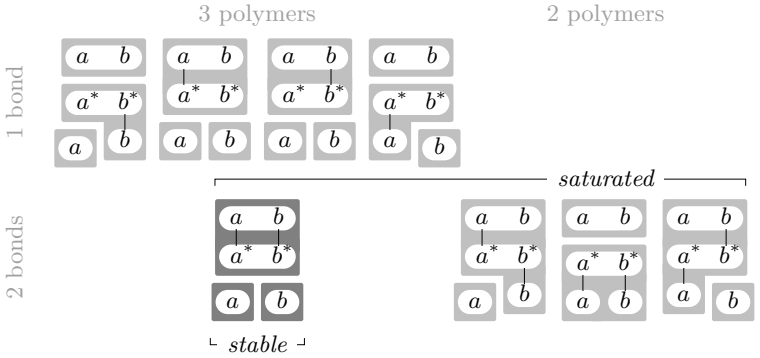

Fig. 1. All configurations of the TBN $T = \{\{a\}, \{b\}, \{a, b\}, \{a^*, b^*\}\}$ with at least one bond. An edge between sites indicates that they pair in that configuration. A panel indicates a polymer. The saturated configurations maximize the number of bonds. The stable configuration further maximizes the number of polymers.

As Figure 1 illustrates, a *configuration* γ of a TBN is a matching among its complementary sites. Two sites *pair* in γ if they are matched. Two monomers *bind* in γ if some of their sites pair. A *polymer* of γ is a connected component with respect to binding. The configuration γ is *saturated* if the matching is maximal. γ is *stable* if it is saturated and no saturated configuration has more polymers.

3 Results

In this work we consider the computational complexity of certain natural questions concerning TBNs and develop a software package to answer those questions [1].

The most basic question is how can we distinguish stable configurations from unstable ones. In the TBN model, the hard part of this is determining whether the system can reconfigure to increase the number of polymers, thereby increasing entropy, without reducing the total number of bonds.

Definition 1. (T, k) *is in* SaturatedConfig *if some saturated configuration of the TBN* T *has at least* k *polymers.* $S(T)$ *is the greatest such* k*.*

Claim 1. SaturatedConfig is in NP-complete.

Claim 2. S has no n^δ factor approximation algorithm running in time polynomial in n for any $\delta < 1/2$ unless P = NP, where n counts the monomers.

In DNA strand displacement cascades, output is usually represented by the release of a previously bound DNA strand. The question of whether releasing the output is favorable corresponds to the problem of deciding whether a given monomer is free in some stable configuration of the TBN model. We show this problem is complete for $\mathrm{P}_{\|}^{\mathrm{NP}}$ (which is P with parallel access to an NP oracle).

Definition 2. $(T, \mathbf{p})$ *is in* StablyFree *iff* $\mathbf{p}$ *can be stably free in* T*.*

Claim 3. StablyFree is in $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete.

Despite these worst-case negative results, we develop a software package accompanying this paper that can answer many questions in practice [1]. The package computes a non-trivial reduction to the boolean satisfiability problem (SAT) which is then passed to a SAT solver. We use it to verify a "counter" tile assembly system [4] and a strand displacement AND gate [9] from prior work.

References

1. https://bitbucket.org/ksbtex/tbnsolverm/.
2. R. D. Barish, et al. An information-bearing seed for nucleating algorithmic self-assembly. *Proceedings of the National Academy of Sciences*, 2009.
3. D. Doty. Theory of algorithmic self-assembly. *Communications of the ACM*, 2012.
4. D. Doty, et al. Thermodynamic binding networks. In *DNA Computing and Molecular Programming: 23rd International Conference*, 249–266. Springer, 2017.
5. L. L. Ong, et al. Programmable self-assembly of three-dimensional nanostructures from 10,000 unique components. *Nature*, 2017.
6. A. Phillips, et al. A programming language for composable DNA circuits. *Journal of the Royal Society Interface*, 2009.
7. R. Schulman, et al. Programmable control of nucleation for algorithmic self-assembly. *SIAM Journal on Computing*, 2009.
8. N. Srinivas, et al. Enzyme-free nucleic acid dynamical systems. *Science*, 2017.
9. C. Thachuk, et al. Leakless DNA strand displacement systems. In *Proceedings of DNA Computing and Molecular Programming 21*, 133–153. Springer, 2015.
10. D. Y. Zhang, et al. Dynamic DNA nanotechnology using strand-displacement reactions. *Nature chemistry*, 2011.