Formal Verification of Network-Based Biocomputation Circuits

Hillel Kugler[1], Till Korten[2], Stefan Diez[2], and Dan Nicolau Jr.[3]

[1] Bar-Ilan University, Israel
[2] TU Dresden, Germany
[3] MolecularSense, UK

Abstract Engineering biological devices to perform computation is of major interest due to the potential to utilize inherent parallelism in biological components to speed computation, construct low energy consuming devices and interface with biological material, opening up potential diagnostic and medical applications. The foundations of a parallel computing paradigm is presented in [6], where a given combinatorial problem is encoded into a graphical, modular network that is embedded in a nanofabricated planar device. Exploring the network in a parallel fashion using a large number of independent, biological agents, allows to solve the mathematical problem. This approach, which is termed network-based biocomputation (NBC), can use low energy compared to standard computing devices and has the potential to be scaled up relying on available nano-design and manufacturing technology. As the complexity of the networks designed grows, and new problem encodings are explored, it is crucial to ensure that the planned devices are indeed correct and do not contain any logical design errors. Towards this goal we have developed a formal verification prototype tool that can establish the correctness of NBC devices or identify design problems, and have applied this framework to new circuits developed in [1]. We envision that simulation and formal verification tools will play an important role in the design process and validation of novel NBC devices.

1 Introduction

Many combinatorial problems of practical importance, such as the design and verification of circuits, the folding and design of proteins, and optimal network routing, require exploring a large number of candidate solutions to check the existence of a satisfying solution. Algorithms in which the time required for solving the problem grows exponentially with the problem size, may be infeasible to run using conventional electronic computers, which operate sequentially. One strategy to solve such problems is to utilize efficient parallel-computation approaches. The foundations of an alternative, parallel computing paradigm is presented in [6], where a given combinatorial problem, the Subset Sum Problem (SSP), was encoded into a graphical, modular network that was embedded in a nanofabricated planar device. Exploring the network in a parallel fashion using a large number of independent, biological agents, then solved the mathematical

problem. This approach can use low energy compared to standard computing devices and has the potential to be scaled up relying on available nano-design and manufacturing technology.

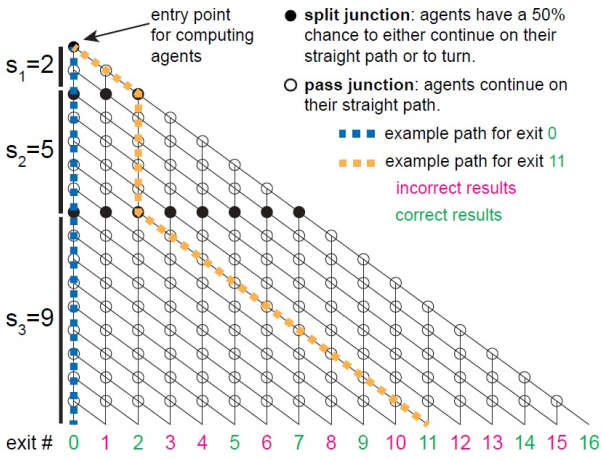

Fig. 1. [6] Computation network for the SSP instance $\{2, 5, 9\}$, the network enables to solve the decision problem and compute the existence of a subset sum for any $T \in \{0 \cdots 16\}$. The agents (molecular-motor-propelled protein filaments) enter the network from the top-left corner. Filled circles represent split-junctions where agents can either continue straight ahead or turn, a stochastic decision that occurs with approximately equal probability. Empty circles represent pass-junctions, where agents continue straight ahead in the direction they were travelling without turning. Moving diagonally down and right at a split-junction corresponds to selecting a number to the subset and adding the number to the sum (numbers 2 and 9 for the path highlighted in yellow). Thus the actual value of the integer potentially added at a split-junction is determined by the number of rows of pass-junctions until the next split-junction. The exit numbers correspond to the target sums T (potential sums with positive solutions) represented by each exit. Correct results for this particular set $\{2, 5, 9\}$ are labelled in green, and incorrect results (where no agents should arrive) are labelled in magenta.

The SSP decision problem asks, given a set of integers $S = \{a_1, a_2, \cdots a_n\}$ and a target sum T, is there a subset $S' = \{a_{i_1}, a_{i_2}, \cdots a_{i_k}\}$ such that the sum of the numbers in S' is exactly equal to T, i.e., $\sum_{j=1}^{k} a_{i_j} = T$. For example, given the set $S = \{2, 5, 9\}$ and the target $T = 7$, selecting the subset $S' = \{2, 5\}$ would sum up to $2 + 5 = T$, while for $T = 13$ there is no subset that would sum up to T. The network encoding of the SSP problem for $S = \{2, 5, 9\}$ is shown in Fig. 1. The main idea is that the network consists of two types of junctions, split-junctions and pass-junctions. A split-junction allows an agent to make a stochastic decision whether to continue moving in the current direction it is travelling or to take a turn, and in the SSP network this decision corresponds

to a decision whether to include a given a_i in the subset or not. A pass junction directs an agent to continue moving in the current direction without turning, thus it will continue moving diagonally and add the number a_i if it was selected in the previous split junction, or continue down until reaching the next split junction, where a decision about the next number a_{i+1} will be made, assuming $i < n$. The SSP problem is an NP-complete problem [5, 3], thus no polynomial algorithm for it is known nor is likely to be found. Since SSP is an NP complete problem, all other problems in NP can be converted to SSP in polynomial time. In recent work we have shown how to encode the Exact Cover (ExCov) problem using network based computation, and are working to manufacture NBC circuits that are larger than the device designed and implemented in [6].

2 Verification Toolset

Formal verification methods are now widely used in the hardware industry [4, 7], enabling to verify the correctness of hardware circuits even before they are manufactured, which can lead to more robust systems and save money, as late detection of errors is extremely costly. As part of the Bio4Comp project [1] we are developing verification algorithms, tools and methodologies to enable similar advantages for Network-Based Biocomputation. Our first version of a toolset enables to verify correctness of SSP and EXCOV problems. The method encodes the networks created in the project as a model within the NuSMV model checker [2], and then uses temporal logic formulas to check if a filament can exit through a specific exit, ensuring e.g., that for an SSP problem only legal sums can be observed as filaments exiting at the required network positions.

References

1. `Bio4Comp` project website, www.bio4comp.org, 2018.
2. A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. Nusmv: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.
3. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge MA, 1990.
4. L. Fix. Fifteen years of formal property verification in intel. *25 Years of Model Checking*, pages 139–144, 2008.
5. M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the theory of NP-Completeness*. W. H. Freeman and Company, 1979.
6. D. V. Nicolau, M. Lard, T. Korten, F. C. van Delft, M. Persson, E. Bengtsson, A. Månsson, S. Diez, H. Linke, and D. V. Nicolau. Parallel computation with molecular-motor-propelled agents in nanofabricated networks. *Proceedings of the National Academy of Sciences*, 113(10):2591–2596, 2016.
7. V. Paruthi. Large-scale application of formal verification: From fiction to fact. In *Formal Methods in Computer-Aided Design (FMCAD), 2010*, pages 175–180. IEEE, 2010.