Engineering and Verifying Requirements for Programmable Self-Assembling Nanomachines (NIER Track)

R. Lutz*† , J. Lutz*, J. Lathrop*, T. Klinge*, E. Henderson‡, D. Mathur‡, and D. Abo Sheasha*
*Department of Computer Science
Iowa State University, Ames, IA 50011 USA
{rlutz, lutz, jil, tklinge, dalia}@iastate.edu
†Jet Propulsion Laboratory
California Institute of Technology, Pasadena, CA 91104 USA
‡Department of Genetics, Development, and Cell Biology
Iowa State University, Ames, IA 50011 USA
{telomere, divita}@iastate.edu

***Abstract*—We propose an extension of van Lamsweerde's goal-oriented requirements engineering to the domain of programmable DNA nanotechnology. This is a domain in which individual devices (agents) are at most a few dozen nanometers in diameter. These devices are programmed to assemble themselves from molecular components and perform their assigned tasks. The devices carry out their tasks in the probabilistic world of chemical kinetics, so they are individually error-prone. However, the number of devices deployed is roughly on the order of a nanomole (a 6 followed by fourteen 0s), and some goals are achieved when enough of these agents achieve their assigned subgoals. We show that it is useful in this setting to augment the AND/OR goal diagrams to allow goal refinements that are mediated by threshold functions, rather than ANDs or ORs. We illustrate this method by engineering requirements for a system of molecular detectors (DNA origami "pliers" that capture target molecules) invented by Kuzuya, Sakai, Yamazaki, Xu, and Komiyama (2011). We model this system in the Prism probabilistic symbolic model checker, and we use Prism to verify that requirements are satisfied, provided that the ratio of target molecules to detectors is neither too high nor too low. This gives prima facie evidence that software engineering methods can be used to make DNA nanotechnology more productive, predictable and safe.**

***Keywords*-Requirements engineering; validation and verification; safety; DNA nanotechnology; molecular programming.**

I. Introduction

Nanotechnology–the control of matter at the nanoscale–promises transformative benefits for medicine, information technology, energy production, and other enterprises of twenty-first century society. The realization of these benefits depends on scaling up the precise nanoscale control of matter. A promising method for such large-scale control is nanoscale self-assembly, the engineering and programming of useful nanomachines that autonomously assemble themselves from molecular components.

The prospect of the programmable self-assembly of nanomachines was enabled by pioneering work of Seeman [1], Winfree [2], and Rothemund [3]. It was Seeman's idea to use the information-processing capabilities of DNA to program short strands of DNA to assemble themselves into specified structures and devices. Winfree showed that self-assembly is Turing universal, i.e., that any computation can be simulated by self-assembly. This implies that self-assembly can be algorithmically directed, whence extremely complex shapes and behaviors can be realized by self-assembly. Doty et al. have recently shown that self-assembly is universal in an even stronger, intrinsically geometric sense [4]. Rothemund introduced DNA origami, a very general method for using short DNA "staples" to cause a long, single-strand DNA "scaffold" (usually the genome of one specific bacteriophage) to fold itself into a desired shape. It is, to date, the most flexible and impressive means of controlling matter at the nanoscale. The prospect of programming molecular devices (e.g., circuits and robots) with dynamic behaviors using DNA strand displacement was raised by work by Yurke et al. [5].

Programmable DNA nanotechnology is a rapidly emerging field. It is highly interdisciplinary, bringing together computer science, molecular biology, biochemistry, and materials science and engineering. While many applications are envisioned, most research is still basic, demonstrating various ways of controlling matter at molecular scales. However, DNA nanotechnology experiments are already so complex that their initial design requires significant use of computer software such as the DSD programming language [6] or caDNAno software [7]. The probabilistic model checker PRISM [8] has been used to verify properties of nanomachines but, to the best of our knowledge, requirements engineering has not been used previously in this domain. It is our contention that the systematic study of requirements and verification for programmable, self-assemblying nanomachines needs to start in this stage. It is especially important to establish a requirements engineering framework well before the envisioned, future deployment of safety-critical applications [9] (e.g., RNA nanomachines

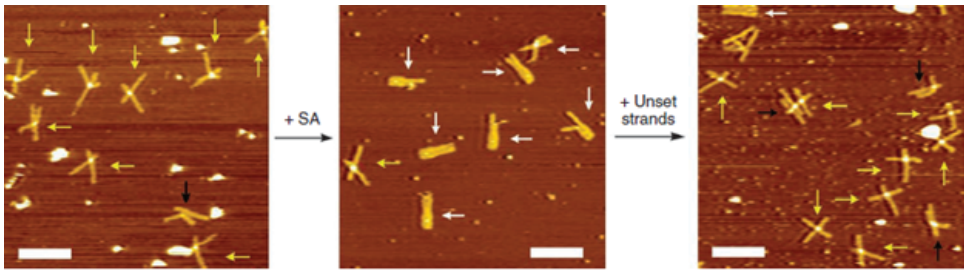

Figure 1. Atomic Force Microscope images of DNA origami pliers: (a) at initiation, DNA pliers are open in the *cross form*, (b) upon adding SA molecules to the solution, DNA pliers close to the *parallel form*, capturing the SA molecules in the pliers' jaws; (c)at reset via addition of *unset strands* DNA pliers re-open, releasing SA molecules. Adapted by permission from Macmillan Publishers Ltd: ref. [12], copyright 2011.

embedded in human cells [10], [11]).

The contribution of this paper is to suggest a discipline by which requirements engineering and formal verification can be performed on programmable, self-assemblying nanomachines. We restrict ourselves here to discussing this in the context of an important, recent development of one such nanomachine, DNA origami pliers [12]. We investigate (1) what it means to specify and validate requirements on programmable DNA self-assemblies and (2) how we can verify formally that a self-assembly satisfies its requirements. While we do not expect an off-the-shelf framework to suffice in this domain, we take as our starting point the identification, evaluation, specification and verification of requirements for DNA origami pliers as a self-assembling, programmable nanomachine, using van Lamsweerde's goal-oriented requirements engineering [13] and the model checking of the derived requirements using PRISM. We describe below our use, results and adjustments of the existing methods to better fit this new domain.

II. Requirements Engineering

The *requirements identification and domain understanding* for this initial work were largely artifact-driven, based on reverse engineering requirements for the nanomachines from papers describing experimental results. Some of us are domain experts, who also served as stand-ins for stakeholders. We gained additional insights into the domain by visits to the wet laboratory. In identifying and refining the requirements, our primary concern was accuracy. The creation of a goal graph and operational model matured the requirements, led to many revisions of domain properties and assumptions, and surfaced additional subgoals. It also provided a way to structure the variety of unfamiliar information that we had.

The *requirements evaluation* found inconsistencies in the vocabulary used due to the interdisciplinary nature of DNA nanotechnology and lab-specific dialects. The evaluation also showed that the functional requirements for the pliers nanomachine were probabilistic. The *requirements specification* constructed iterative versions of the goal graphs that were increasingly precise and accurate. The plan to formally verify the requirements led to a better informal representation, as well. We simultaneously built a formal state machine model in PRISM, and formalized a set of key requirements in it. *Requirements validation and verification* by simulation and model checking helped assure that these requirements were satisfied in the model, as described below.

The nanomachines to be built are complex, asynchronous, probabilistic, and resource constrained. They have both functional and nonfunctional requirements. Functional requirements state what the self-assembly shall do. The most important functional requirements are geometric, i.e., specifying the shape or structure that the assembly shall achieve, or behavioral, i.e, specifying what the assembled shape shall do.

DNA origami pliers. We consider a DNA origami structure that can detect the presence of a target molecule in a solution [12]. The pliers are a nanomachine because they open and close autonomously (i.e., change their shape) depending upon the presence or absence of a target molecule (in this case, SA-streptavidin tetramer) in their environment. They thus can serve as monitors to "diagnose" whether SA is present. A future such application might be to detect harmful molecules within human cells. As shown in Fig. 1(a), the DNA origami structure is programmed to assemble in the shape of "pliers" in the open position. (Each yellow cross pointed to by an arrow is an open pliers.) The pliers are programmed such that, once assembled, they will capture a particular molecule of interest by closing that molecule in their jaws, Fig. 1(b). They should then release the molecule when "unset strands" of DNA are added to the solution. Observation through an atomic force microscope (AFM) of the pliers shows whether the molecule of interest is present in the solution.

There is a key difference between goal modeling for this domain and for others in that we need to distinguish the goals for the entire system from the goals for the individual pliers. The high-level goal is to detect the target molecule. The assigned agents to achieve this are the pliers. There are thus a huge number of agents, roughly on the order of a nanomole (6×10^{14}) of them. Each of these agents has a significant probability of failing to achieve its goal.

The desired behavior for the system is achieved if a

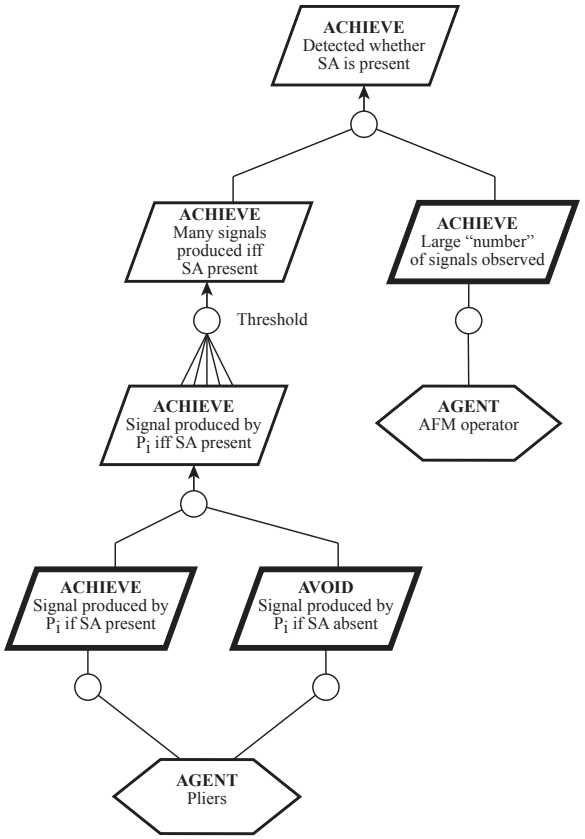

Figure 2. Goal graph

sufficient number of the agents achieve their assigned, nearly identical subgoals. We thus found it useful to augment the AND/OR goal diagrams to allow goal refinements that are mediated by threshold functions, rather than ANDs or ORs. (While a threshold function can logically be realized as an OR of ANDs, this is exponentially larger and unintuitive.) We thus add a new kind of node, the THRESHOLD node, with an associated *threshold ratio*, *TR*. The meaning here is that the goal is achieved if the number of subgoals that are achieved exceeds the fraction *TR* of the total number of subgoals of the THRESHOLD node. Note that threshold nodes are not probabilistic goals [14]. They depend not on the probability that something *might* happen, but on the multiplicity of times that it *does* happen.

The goal graph, shown in Fig. 2, has the top-level goal, Achieve[Detected Whether SA Is Present]. This goal is refined in the diagram into subgoals Achieve[Many Signals Produced iff (if and only if) SA Is Present] and Achieve[Large Number of Signals Observed]. The latter subgoal is assigned to a human agent, the AFM operator, who images a sample of the solution and manually counts the state of the pliers. The former subgoal is achieved provided that a sufficient number of individual pliers each produce a signal if and only if SA is present. The signal is the change to the closed shape.

There is no cooperative or swarm behavior here, nor do the sensors form a network. However, the requirements for programming pliers to capture a SA molecule are very similar, but not identical to, the requirements for capturing other molecules. It appears likely that such highly similar self-assemblies may be usefully considered as a product line [15].

It is also worth mention that the process of constructing the goal graph alerted us to a failure mode that we had not considered, namely, that SA molecules might bind to both jaws of a pliers before they can close. PRISM verification indicates that this can happen often enough to prevent detection if the concentration of SA is sufficiently high.

III. Verification

We created a continuous-time Markov chain model in PRISM of the reactions described in [12] that model the detection of target molecules. We then specified properties that define correct operation and used PRISM to verify that these properties are satisfied. We used a Markov chain, shown in Fig 3, and the PRISM software tool to model the behavior of a large number of pliers existing in solution with SA molecules or unset strands. The state of the model depended on the state of the individual pliers in the system. Each pliers had three primary states: open, closed, and anti-parallel. Pliers in open form could loosely capture an SA molecule in the left or the right side of its jaw and fully capture it by closing while it is caught. With unset strands in the system, it was possible for them to displace the gripping surface on one or both of the jaws of the pliers. The number of gripping surfaces present contributed to the state of the system and increased the complexity of the Markov chain.

Three of the requirements derived above were written as properties within PRISM in order to verify their correctness: (1) ensuring capture when the target molecule, SA, is present, (2) avoiding false positives when the target molecule is not present, and (3) avoiding pliers in error states (such as one SA in each side of its jaw). These properties were implemented in PRISM's property specification language and verified using PRISM's implementation of the Gauss-Seidel method for solving linear systems of equations.

We also found that as the concentration of SA is increased, the expected value of total captures decreases. This is illustrated in Fig 4. In this graph, the y-axis shows the ratio of the number of pliers in a certain state and the x-axis shows the number of SA molecules that are in the system. The blue series represents the expected number of pliers with successful captures, and the green series represents the expected number of pliers resulting in an error state.

IV. Conclusion

We have seen that adding threshold-mediated goal refinement to van Lamsweerde's goal models enables requirements engineering to be used—for the first time to

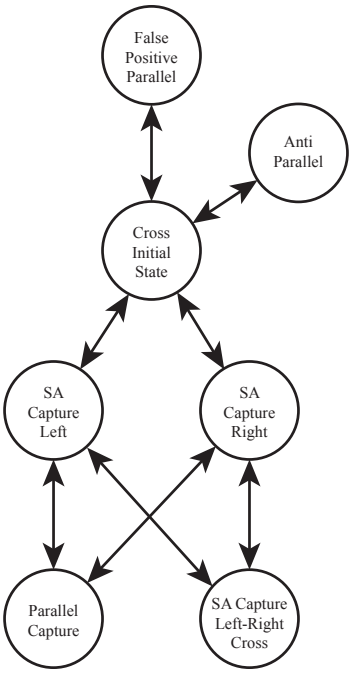

Figure 3. PRISM state model

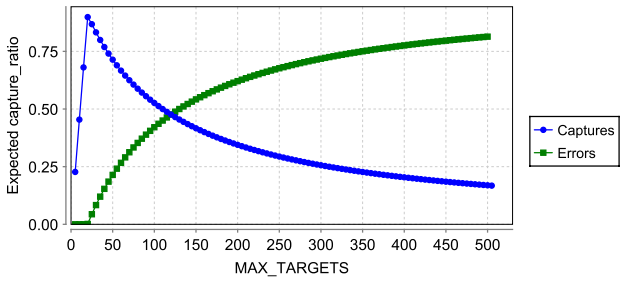

Figure 4. Error Graph

the best of our knowledge—in the emerging field of DNA nanotechnology. We have illustrated this approach by applying it to the DNA origami pliers molecular detection system, thereby obtaining requirements that we have verified using the Prism probabilistic symbolic model checker. We believe that this work provides prima facie evidence that software engineering methods can be used to make DNA nanotechnology more productive (e.g., by improved and more automated design of wet-lab experiments), predictable (e.g., by improved methods for reasoning about the behaviors of self-assembling, programmable systems operating in kinetic molecular environments), and safe (e.g., by analyzing the embedded systems obtained when programmable nanodevices are deployed in living cells). Only a great deal of research by many investigators will turn this prima facie case into a clear scientific demonstration.

ACKNOWLEDGMENTS

We thank Samik Basu for useful discussions. This research was supported by NSF grants 1143830, 0916275, and 0652569. Part of this work was performed while the first two authors were on sabbatical at Caltech.

REFERENCES

[1] N. Seeman, "Nucleic acid junctions and lattices," *Journal of Theoretical Biology*, vol. 99, pp. 237–247, 1982.

[2] E. Winfree, "Algorithmic self-assembly of DNA," Ph.D. dissertation, California Institute of Technology, 1998.

[3] P. W. K. Rothemund, "Folding DNA to create nanoscale shapes and patterns," *Nature*, vol. 440, pp. 297–302, 2006.

[4] D. Doty, J. H. Lutz, M. J. Patitz, R. T. Schweller, S. M. Summers, and D. Woods, "The tile assembly model is intrinsically universal," *ArXiv e-prints*, Nov. 2011.

[5] B. Yurke, A. J. Turberfield, J. Allen P. Mills, F. C. Simmel, and J. L. Neumann, "A DNA-fuelled molecular machine made of DNA," *Nature*, vol. 406, pp. 605–608, 2000.

[6] A. Phillips and L. Cardelli, "A programming language for composable DNA circuits," *Journal of the Royal Society Interface*, vol. 6, pp. S419–S436, 2009.

[7] S. M. Douglas, A. H. Marblestone, S. Teerapittayanon, A. Vazquez, G. M. Church, and W. M. Shih, "Rapid prototyping of 3D DNA-origami shapes with caDNAno," *Nucleic Acids Research*, vol. 37, no. 15, pp. 5001–5006, 2009.

[8] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV11)*, vol. 6806. Springer, 2011, pp. 585–591.

[9] N. G. Leveson, *Safeware: system safety and computers*. New York, NY, USA: ACM, 1995.

[10] K. Rinaudo, L. Bleris, R. Maddamsetti, S. Subramanian, R. Weiss, and Y. Benenson, "A universal RNAi-based logic evaluator that operates in mammalian cells," *Nature Biotechnology*, vol. 25, pp. 795–801, 2007.

[11] S. Venkataraman, R. Dirks, C. Ueda, and N. Pierce, "Selective cell death mediated by small conditional RNAs," in *Proc Natl Acad Sci USA*, vol. 107, no. 39, 2010, pp. 16 777–16 782.

[12] A. Kuzuya, Y. Sakai, T. Yamazaki, Y. Xu, and M. Komiyama, "Nanomechanical DNA origami 'single-molecule beacons' directly imaged by atomic force microscopy," *Nat Commun*, vol. 2, 2011/08/23/online.

[13] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Chicester, England: Wiley, 2009.

[14] E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *SIGSOFT FSE*, 2004, pp. 53–62.

[15] J. Dehlinger and R. R. Lutz, "Gaia-PL: A product line engineering approach for efficiently designing multiagent systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 4, p. 17, 2011.