A Compiler for Programming Molecular Robots

Inbal Wiesel, *Noa Agmon* and *Gal A. Kaminka*
Computer Science Department
Bar Ilan University, Israel
Corresponding author: galk@cs.biu.ac.il

1. Introduction. Molecular robotics are a promising approach for biomedical applications. Molecular robots (nanobots) can operate inside a living body [2, 3, 1, 11], carrying out molecular actions, such as releasing a molecular payload only under some environmental conditions or shielding the body from toxic payloads [4]. If used as a platform for drug delivery, a nanobot can, in principle, overcome many of the safety issues, as drugs are released only in the presence of their targets. Nano-scale devices—including but not limited to nano-robots—may also be used to carry out computations [13], but our focus here is on robotic tasks (such as drug delivery) rather than arbitrary computations.

Currently, every nanobot must be designed by experts, for the specific task: medical expertise must meet nanobot design expertise. As procedures grow in complexity, the challenge is exacerbated: nanobot developers mix different types of nanobots—each type specifically tailored—in nanobot cocktails (swarms), such that the medical outcome emerges out of the interactions of the various nanobots [10].

Recent advances have begun to explore generic nanobot arch-types, which can be "programmed" (specialized) in specific ways. Recently developed nano-particles [11] serve as an example. These nanobots are built from a nanometer-scale gold bead, to which various DNA strands can be attached, e.g., to bind with specific biomarkers. The DNA-based *clamshell* [4] and *walker* [8] nanobots are other examples.

The development of programmable nanobot arch-types offers an opportunity to consider tools for programming nanobot cocktails. Inspired by modern software development environments, which separates high-level programming languages from specific CPU details, we aim to allow medical professionals to directly program treatments in a *Athelas*, a medication programming language. the design of Athelas is motivated by the success of rule-based systems at capturing expert knowledge [6, 7]. A compiler (*Bilbo*) translates Athelas programs to nanobot specifications, which implement the program. The compiler relies on a library of generic nanobot arch-types, and specializes them to create the specific roles needed for the swarm.

We believe this separation between medical expertise and nanobot design expertise can significantly accelerate the development of new medical treatments relying on nanobot technology: Medical experts will program treatments. Molecular roboticists will develop generic nanobots. And compilers will synthesize swarms of nanobots that carry out the programs, with performance and safety guarantees.

2. The Athelas Language. We consider nanobot tasks such as moving compounds between locations in the body, picking compounds (by molecular binding), or exposing (and sometimes releasing) them in diseased areas. To specify tasks, we adopt a *rule-based programming* paradigm, in which programs are specified by sets of rules that are continuously considered in parallel, against changing conditions [5]).

Each rule has four clauses. The `Initialize` clause specifies the set of payloads to be built into the drug when it is injected (i.e., before any action is taken). The `When` and `Until` clauses are each composed of a set of tests, e.g., pH level or concentration of a specific chemical in specific location. The `When` tests must hold in order for the drug to become activated (here, when the concentration of `Y` in the vicinity of `T` is above $5mol/m^3$). The `Until` terminate the activity of the drug. The `Actions` clause contains the actions to be executed when the drug is active, e.g., `pick, drop, protect, expose, disable` and other actions.

3. The Bilbo Compiler. The Bilbo compiler takes two inputs: an Athelas program, and a library of generic robot types (with defined ways of parameterizing them, including parameterizable preparation

protocols). It then synthesizes a specification for a heterogeneous swarm of specialized nanobots, which would carry out the program, once deployed. The output specification for each specialized robot includes a specialized preparation protocol.

The compilation process is done in two phases. A front-end phase consists of the lexical and syntax analyzers, generates finite state machines (FSMs) representing the rules. The back-end phase then transforms such FSMs into a final nanobot swarm specification (recipe). This is done by a graph-rewriting approach, with specific operators for merging, expanding FSM transitions and states, and rejecting incorrect paths. At the end of this process, an AND/OR graph emerges, which represents all possible nano-swarms (cocktails) that can carry out the program. The compiler uses the AO* [9] algorithm to determine an optimal AND/OR path in the graph, which corresponds to a specific heterogeneous swarm, made of specialized nanobot arch-types and their preparation protocols.

4. Validation Elsewhere [12], we reported on experiments where the compiler was used to generate cocktails with 2–5 nanobot types, made of clamshell and nano-particle arch-types. While we continue our work on this front, we want to report here on other validation work we are carrying out.

In particular, since the generated nanobot swarms are intended to one day serve in biomedical applications, we also consider safety and performance guarantees. We have proven that the main compilation algorithms are *complete*, meaning that all possible cocktail alternatives are found. We have additionally proven that these algorithms are *sound*, meaning that any cocktail found by the system is indeed a valid one (in the sense of matching the Athelas program).

References

[1] Y. Amir, E. Ben-Ishay, D. Levner, S. Ittah, A. Abu-Horowitz, and I. Bachelet. Universal computing by DNA origami robots in a living animal. *Nature Nanotechnology*, 9(5):353–357, May 2014.

[2] A. Cavalcanti, B. Shirinzadeh, T. Fukuda, and S. Ikeda. Nanorobot for brain aneurysm. *International Journal of Robotics Research*, 28(4):558–570, 2009.

[3] L. Dong and B. Nelson. Tutorial - robotics in the small part ii: Nanorobotics. *Robotics Automation Magazine, IEEE*, 14(3):111–121, Sept 2007.

[4] S. M. Douglas, I. Bachelet, and G. M. Church. A logic-gated nanorobot for targeted transport of molecular payloads. *Science*, 335(6070):831–834, Feb 2012.

[5] A. Gupta, C. Forgy, A. Newell, and R. Wedig. Parallel algorithms and architectures for rule-based systems. *SIGARCH Computer Architecture News*, 14(2):28–37, May 1986.

[6] F. Hayes-Roth. Rule-based systems. *Communications of the ACM*, 28(9):921–932, Sep 1985.

[7] A. A. Hopgood. *Intelligent Systems for Engineers and Scientists*. CRC Press, 2001.

[8] K. Lund, A. J. Manzo, N. Dabby, N. Michelotti, A. Johnson-Buck, J. Nangreave, S. Taylor, R. Pei, M. N. Stojanovic, N. G. Walter, E. Winfree, and H. Yan. Molecular robots guided by prescriptive landscapes. *Nature*, 465(7295):206–210, 2010.

[9] N. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing, Palo Alto, CA, 1980.

[10] E. Ruoslahti, S. N. Bhatia, and M. J. Sailor. Targeting of drugs and nanoparticles to tumors. *Journal of Cell Biology*, 188(6):759–768, 2010.

[11] G. Y. Tonga, Y. Jeong, B. Duncan, T. Mizuhara, R. Mout, R. Das, S. T. Kim, Y.-C. Yeh, B. Yan, S. Hou, et al. Supramolecular regulation of bioorthogonal catalysis in cells using nanoparticle-embedded transition metal catalysts. *Nature chemistry*, 7(7):597–603, 2015.

[12] I. Wiesel, G. A. Kaminka, G. Hachmon, N. Agmon, and I. Bachelet. Late-breaking: First steps towards automated implementation of molecular robot tasks. In *DNA Computing (DNA-21)*, 2015.

[13] E. Winfree, X. Yang, and N. C. Seeman. Universal computation via self-assembly of dna: Some theory and experiments. In *DNA based computers 2*, volume 2, pages 191–213. Amer Mathematical Society, 1999.