論文の内容の要旨

Assisted Design of DNA Computing Systems: the DNA Toolbox and Beyond (DNAコンピューティングシステムの設計支援: DNAツールボックスとその拡張)

氏名 Aubert Nathanael Yann Ivan
オベル ナタナエル ヤン エバン

In the recent years DNA computing has shown promising results with potential applications in a wide range of fields, such as medicine and smart material engineering. DNA computing leverages the well understood biochemistry of DNA to encode calculus and perform computation, with the goal of enabling computation in inconvenient places, such as inside the human body. However, most ideas often stay at the stage of proof of concept, without making it to the marketable product level.

The reason for this is not a lack of interest from designer, but lies instead in the fact that such products have a structural complexity that is beyond the tools currently available in the field. Even though interactions such as DNA-DNA or DNA-enzyme are reliably predictable, they cause counter-intuitive behaviors, such as Michaelis-Menten enzyme saturation. For this reason, even simple systems may require a long trial-and-error design process.

The goal of this thesis was to develop means to ease the design process of such DNA computing systems. This relies mainly on three complementary approaches: module design, development of specific computer assisted design (CAD) tools, and search of interesting patterns evolved through artificial evolutionary algorithms. Those different aspects are each used to simplify a particular aspect of the design process.

The first step was to create a new module that could be easily integrated to popular DNA computing paradigms. The role of this module is to perform a delay or join (in the traditional computer science meaning) operation. Indeed, one major design problem is that intermediary reaction products can react with any other part of a system, not only their intended target. While careful sequence design can avoid unwanted interactions up to a certain point, it still leaves the problem of leaks: during a given computation step, the wrong output may be present as a temporary product, before the correct one is finally released. If the incorrect output is allowed to interact with downstream elements of the system, errors will propagate. This problem is akin to concurrency problems in computer science, and can be resolved using the delay module by ensuring that a given computation is finished before allowing the next step to happen. More specifically, the delay module relies on timer strands capturing the targeted sequence and being then catalytically consumed by a DNA delay gate. Timer strands are present in excess with respect to their target and are designed so that the target is only freed once no timer strand remains, ensuring the delay. This approach was confirmed both by simulation and experiment.

The second step was to develop an efficient simulator to help the designer prototype systems. Indeed, while actual experimentation is necessary to check the correctness of a system's behavior, the cost in time and resources prevents its use as a convenient way to design. Instead, simulation allows the user to refine the system and check for flaws. We targeted the DNA toolbox, a set of three modules (activation, inhibition and autocatalysis) used for the rational design of DNA computing systems. Its simple modules allow a straightforward representation of systems as a graph. Moreover, those modules can be easily derived into a mathematical model close to their actual *in-vitro* behavior. This means that systems

designed with this CAD software have reasonable chances of behaving in the same fashion in an actual implementation. The software also offers to export designed systems in the Synthetic Biology Mark-up Language format, allowing the user to use other design tools, such as Copasi.

The third step was to take the opposite design approach: since it is hard to come up with a structure performing a given task, it is instead possible to look at the possible behaviors that arise from small structures. To direct this search, we used an *in silico* evolutionary approach. Systems were asked to "play" a simple game of rock-paper-scissors, with a fitness based on their success rates. By performing an analysis of the evolution of well-performing systems, it was possible to highlight multiple substructures with specialized roles. While most of those patterns were specific to the problem at hand, some can be reused in different contexts. This shows another positive point of this approach: more and more building blocks are found over time, making the designing process easier.