Simulation of DNA-Computing

Eike Best, Heiko Burchard, Hans Fleischhack, André Hackmann, Stefan Kühnapfel, Ernst Kretschmann, Astrid Rakow

Carl von Ossietzky Universität Oldenburg, D-26111 Oldenburg

This poster describes an implemented client/server architecture supporting fast simulation of DNA computing. Users may create virtual DNA laboratories on (slow) client computers by communicating with a (fast) server machine using a script language. Efficient algorithms simulating standard (and more) DNA operations are implemented on the server.

The system is based on an object-oriented, re-usable and extendable architecture. Basic elements are DNA strands and enzymes. Basic operations are various reactions between instances of these classes in a test tube. The class DNA strand supports recognition points and markings. The script language allows straight line code but hides conditional and iterative constructs. Such control is left to the user who may take any course of action depending on the outcome of prior experiments. Experiments are transmitted to the server by a TCP/IP connection; results of experiments are transmitted back from the server to the (correct) client in the same way. The interface between clients and server has the functionality of a scanner, a parser and an interpreter. It is concise and extendable, and its class structure conforms with that on the server side.

Efficient DNA simulation is faced with two severe problems. First, DNA computing relies on efficient use of parallelism, which may not be present on the server. Secondly, genetic methods on which DNA computing is based are subject to rapid development, and it is not foreseeable which new enzymes or reaction mechanisms will be feasible even in the near future. The second problem calls for flexible modelling and implementation, which may impact adversely on speed. So does the first problem. It is all the more important to find efficient data representations and algorithms within the remaining degrees of freedom.

Our solution is to develop an abstraction of genetic sequences in terms of data structures that can be characterised as follows. Every base (pair) is encoded by four bits. Bit 3 equals 1 for single bases and 0 for base pairs. Bit 2 indicates the orientation of single bases. Bits 0 and 1 indicate the base used; they are such that if two complementary bases are added, a (binary) value of 11 results. This representation allows efficient algorithms to be implemented. Every sequence transmitted to the server by some user is first translated into such a representation. On it, DNA operations such as the application of enzymes can be simulated. Operations are then carried out as specified by the user, and the result is translated back into the input format. The code suffices for representing all the basic bases and base pairs. We have provided some freedom for future developments. For instance, an escape symbol can be represented, making the coding extendable (albeit at the expense of efficiency). Also, sequences can be provided with information such as the linking of methylen groups onto specific nucleotides.