Towards an in-vitro Implementation of a Universal Distributed Splicing Model for DNA Computation

Thomas Hinze and Monika Sturm

Dresden University of Technology, Germany
Department of Theoretical Computer Science
e-mail: {hinze,sturm}@tcs.inf.tu-dresden.de
www: http://wwwtcs.inf.tu-dresden.de/dnacomp

Abstract

Emphasizing a combination of recent developments in computer science with molecular bioengineering, a special distributed splicing system (TT6) is proposed. This unconventional model for computation features by a possibility for an in-vitro implementation in the laboratory as well as by a mathematical exact description. The Dresden DNA Computation Group decided to implement such a system on biohardware and optimized all relevant model parameters and components with respect to this objective.

1 Introduction

DNA computing as an unconventional architecture for computing will only convince if a implementation of a biocomputer in practice succeeds. Research activities in this field serve both aspects, theoretical contributions about models for DNA computation, their computational power and optimization as well as real experiments using molecular biological processes. The vision in DNA computing consists in establishing an applicable universal biocomputer. This biocomputer should feature by its computational speed using massive data parallelism. Furthermore this kind of computer will be able to store large amounts of data with a much higher density than electronic memory circuits and requires only a fraction of energy. It should also be mentioned that all components of biocomputers can be recycled completely.

This paper introduces an approach towards a universal biocomputer based on distributed splicing with 6 test tubes with respect to its in-vitro implementation as an issue of DNA computing research in Dresden, Germany. The Dresden DNA computation group deals with laboratory-practical approaches and models for DNA computation closed to observable molecular biological processes. First, a DNA based algorithm to solve the NP-complete knapsack problem with natural object weights was developed and implemented repetitively in the laboratory. In parallel to these experimental studies the laboratory-like DNA computing model DNA-HASKELL[5] was conceived based on molecular biological processes observed in detail including some side effects that can occur indeed. DNA computing models must be computational complete. DNA-HASKELL owns this property proved by simulation of selected conventional universal models for computation.

There is a variety of different DNA computing models which are characterized by a high abstraction level and by a clear formal model description. Between these models and an according implementation in the laboratory a gap exists that has to be discussed. Simulating those DNA computing models using the laboratory-like DNA-HASKELL could be a promising approach to fill this gap and to combine the advantages of different models.

2 Selection of an Appropriate Model

Which well-known model for DNA computation is most suitable to be implemented on biohardware? The decision thereover considers many aspects that can be divided into three classes concerning used data structures, the control mechanism for the sequence of basic steps, and special features with respect to the molecular biological toolbox. Oppositional properties inside these aspects form a classification that supports the selection. Preferred models are configurable in a way that they can behave flexibly and they have an inherent adaptability depending on parameters. All details of a model specification have to be transformable into available resources, materials, methods, and laboratory techniques.

Beside the composition of the DNA material, the data structure aspect contains information whether the model is restricted or unrestricted, whether it uses single or multiple data, and whether or not multisets are considered. The restriction of a model coincides with the one-pot ability: If consecutive operations are performed inside the same test tube, intermediate contents are destroyed (restricted). Otherwise, unrestricted models allow aliquots (copies) of test tube contents. Single data is a property of models which use a homogeneous variety of identical DNA molecules. All these molecules are processed in the same way inducing a high redundancy by strand copies. Massive data parallelism is based on multiple data caused by a variety of different DNA strands in the same test tube. Linear DNA can be generated and analyzed easily, circular DNA is available by plasmids from bacteria. Other forms of nonlinear DNA are difficult to handle.

The control mechanism defines the degree of parallelism between operations (single/multiple instruction) as well as the deterministic or nondeterministic behavior. Sequences of operations can be repeated exactly using deterministic models. In contrast, nondeterministic models are faster in some cases, and heuristical algorithms can be applied.

With respect to implementation details in the laboratory, important criteria express whether or not a model requires enzymatic reactions, whether or not DNA strands are fixed on a surface, how the initial input DNA is generated, which possibilities exist to visualize the final result and the probability for side effects and their suppression, for example.

Filtering models, the sticker model, the Turing machine by Rothemund and splicing systems embody classes of DNA computing models with different underlying concepts and ideas[2]. Splicing systems avoid the brute force strategy. They seem to be flexible enough to adapt the most of the properties mentioned above by varying certain system parameters. Their core, the splicing operation, is directly derived from recombinant techniques. Motivated by these facts, the implementation focusses a splicing system.

Splicing systems are established to reach universal computational power by generating the class of recursive enumerable languages ($\mathcal{RE}$)[3], [6]. Model parameters like number of test tubes, axioms, rules, strand duplicates as well as filter pattern and used DNA structure have to meet requirements for the implementation in the laboratory. For instance, extended Head systems need either an infinite set of axioms or an infinite set of splicing rules to generate $\mathcal{RE}$ resulting in an infinite number of DNA strands and restriction enzymes. A further approach based on multisets leads to the necessary to determine the number of strand duplicates with high accuracy. The recent state of the art in molecular bioengineering can not meet these requirements completely. Therefore other extensions of splicing systems were sought for a practicable possibility. The introduction of distributed splicing systems with n test tubes[1] seems to be a successful way. The number of test tubes, axioms, and splicing rules must be balanced together with an appropriate filter pattern and distribution mechanism resulting in a lab-practicable compromise. The proposed system TT6 tries to accept the challenge.

3 Model Overview and Adaption

The Distributed Extended Head System with 6 Test Tubes (TT6 for short) is able to generate $\mathcal{RE}$ based on an arbitrary Chomsky type 0 grammar G achieving computational completeness[7].

TT6 is composed by a finite set $\mathcal{V}$ of alphabet symbols (containing the sets of nonterminal and terminal symbols from the underlying grammar G and some auxiliary symbols) supplemented by test tubes T_i, $i = 1, \ldots, 6$. Each test tube $T_i = (\mathcal{A}_i, \mathcal{R}_i, \mathcal{F}_i)$ is called a component with a finite set of axioms $\mathcal{A}_i$, a finite set of splicing rules $\mathcal{R}_i$, and a finite set of filter patterns $\mathcal{F}_i$. Test tube T_6 acts as a final tube and collects exactly those strings that represent words of the language $\mathcal{L}(G)$. The test tubes T_1 until T_5 perform iterated loops in parallel. Each iterated loop consists of the consecutive steps splicing operation, filtering, and distributing.

The splicing operation[4] forms the core of all types of splicing systems and embodies an abstract formal emulation of DNA recombinant techniques cut with restriction enzymes (digestion) and ligation. The description of the splicing operation on words of formal languages leads to a generalization of the effect that is caused by digestion and ligation. The generalization suppresses certain DNA strands resp. words than can really occur as side effects. The most frequent unwanted effects are incomplete digestion and ligation as well as production of DNA fragments with false composition by ligation. Additional extractions and labelings prevent these side effects and allow a description of the splicing operation in an experimental convincing way. During each iterated loop the splicing operation is performed at most once per test tube.

After splicing, the subsequent filtering step prepares copies of those strings that have to be distributed. Every test tube T_i, $i = 1, \ldots, 5$ provides separately exactly those strings that will be moved into other test tubes. To do so, T_i evaluates all filter pattern $\mathcal{F}_j$, $j = 1, \ldots, 5$, $j \neq i$. Those strands that are transmitted into other tubes are removed from the producing tube T_i if they do not match its own filter pattern $\mathcal{F}_i$. Each $\mathcal{F}_i$ describes those strings that are moved from other test tubes into T_i. The filter patterns are constructed in a way that each filtering process can be implemented by polymerase chain reaction (PCR). That means, the filter patterns are preferrably described by leftmost and rightmost word ending symbols serving as primers. The PCR leads to an amplification of DNA strands and increases DNA concentrations inside the test tubes.

The subsequent distributing step exchanges the strings prepared by filtering between the test tubes. A union cascade in each iterated loop merges intermediate products and supplies also the final tube.

The steps splicing operation, filtering, and distributing forming the iterated loop are executed consecutively. After distributing, the next round of splicing starts. The number of iterated loops is not limited.

4 Conclusions

This paper implies a proposal to the discussion about distributed splicing systems. The objectives leading to the development of TT6 include the compliance with a description of $\mathcal{RE}$ by a constant number of test tubes, axioms, and splicing rules, by a nonextended DNA structure, and by an efficient derivation of complexity theoretical relevant system parameters directly from the grammar. TT6 is constructed with strong regard to a practicable implementation in the laboratory. The distribution of DNA strands between test tubes is organized in a way that minimizes the number of transferred DNA strands. Repetitive amplifications counteract the decrease of DNA concentration caused by distribution. Beyond only few strand duplicates are necessary to perform all filtering and distributing processes. The number of DNA double strands that have to be available initially is equal to the number of axioms. The operations forming TT6 are based on observable processes in the laboratory.

References

[1] E. Csuhaj-Varj, L. Kari, G. Păun. Test tube distributed systems based on splicing. Computers and AI, vol. 15(2–3), p. 211-232, 1996

[2] M.J. Daley, M.G. Eramian. Models of DNA Computation. Term Report CS881b, L. Kari, Instructor, 1998

[3] R. Freund, L. Kari, G. Păun. DNA computing based on splicing: the existence of universal computers. Theory of Computing Systems, vol. 32, p. 69-112, 1999

[4] T. Head. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. Bulletin of the Mathematical Biology, vol. 49(6), p. 737-759, 1987

[5] T. Hinze, M. Sturm. A universal functional approach to DNA computing and its experimental practicability. Prel. Proceedings of DNA6, Leiden, NL, 2000

[6] G. Păun. SPLICING – a challenge for formal language theorists. Journal of Automata, Languages and Combinatorics, vol. 4, no. 1, p. 3-16, 1999

[7] M. Sturm, T. Hinze. Distributed Splicing of $\mathcal{RE}$ with 6 Test Tubes. Prel. Proceedings of WS on Multiset Processing, Curtea de Arges, Romania, 2000