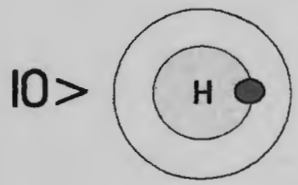
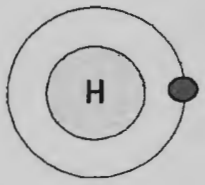


5'----G-A-A-T-T-C----3'
3'----C-T-T-A-A-G----5'



THEORIETAG 2000

mit Workshop

NEW COMPUTING PARADIGMS: MOLECULAR COMPUTING and QUANTUM COMPUTING

25. - 27. September 2000
Wien, Österreich

Edited by
RUDOLF FREUND

TECHNISCHE UNIVERSITÄT WIEN
Institut für Computersprachen

Proceedings

THEORIETAG 2000

mit Workshop

NEW COMPUTING PARADIGMS: MOLECULAR COMPUTING and QUANTUM COMPUTING

25. - 27. September 2000
Wien, Österreich

Edited by
RUDOLF FREUND

TECHNISCHE UNIVERSITÄT WIEN
Institut für Computersprachen

Title: Theorietag 2000 mit Workshop
"New Computing Paradigms:
Molecular Computing and Quantum Computing"

Published by: Technische Universität Wien
Institut für Computersprachen
Favoritenstr. 9, A-1040 Wien, AUSTRIA

Edited by: Rudolf Freund
September 2000

Printed by: Druckerei Berger
Wiener Str. 80, A-3580 Horn, AUSTRIA

Published in September 2000

ISBN 3-85028-325-9

Vorwort

Seit 1991 wird von der GI-Fachgruppe 0.1.5 *Automaten und Formale Sprachen* jährlich der Theorietag mit der jährlichen Fachgruppensitzung veranstaltet. Die Serie begann 1991 in Magdeburg, wurde 1992 in Kiel, 1993 in Dagstuhl, 1994 in Herrsching, 1995 in Schloss Rauischholzhausen, 1996 in Cunnersdorf, 1997 in Barnstorf, 1998 in Riveris und 1999 in Schauenburg-Elmshagen fortgesetzt. 2000 findet der Theorietag erstmals außerhalb Deutschlands in Österreich an der Technischen Universität Wien statt. Damit soll auch eine Öffnung gegenüber dem Osten signalisiert werden.

Die Tradition beibehaltend, wird ein eintägiger Workshop dem Theorietag vorangestellt, der heuer unter dem Thema *New Computing Paradigms: Molecular Computing and Quantum Computing* steht. Die Vortragenden sind

zum Thema Molecular Computing

- Erzsébet Csuhaj-Varjú (Budapest, Ungarn)
- Nataša Jonoska (Tampa, USA)
- Gheorghe Păun (Bukarest, Rumänien)

zum Thema Quantum Computing

- Jozef Gruska (Bratislava, Slowakei)
- Rusiņš Freivalds (Riga, Lettland)
- Karl Svozil (Wien, Österreich)

Die Vorträge werden im Zemanek-Saal der Technischen Universität gehalten; dieser Vortragsraum wurde erst 1999 nach Heinz Zemanek benannt.

Dem Bundesministerium für Bildung, Wissenschaft und Kultur gebührt Dank für die finanzielle Unterstützung des Theorietages.

Wir wünschen allen Teilnehmerinnen und Teilnehmern einen interessanten und ertragreichen Theorietag und einen schönen Aufenthalt in Wien.

Wien, im September 2000

Rudi Freund und sein Team

Inhaltsverzeichnis

TEIL B: Beiträge zum Theorietag

TEIL A

WORKSHOP

New Computing Paradigms

Kapitel 1

Molecular Computing

On Language-theoretic Aspects of Watson-Crick Complementarity: Models and Results *

Erzsébet Csuhaj-Varjú
Computer and Automation Institute
Hungarian Academy of Sciences
H-1111 Budapest
Kende u. 13-17
Hungary
E-mail: csuhaj@sztaki.hu

Abstract

Watson-Crick complementarity is a fundamental concept in DNA computing, inspired by the phenomenon of the formation of double DNA strands. In this paper we give a brief review on two important computational paradigms based on Watson-Crick complementarity: Watson-Crick automata and Watson-Crick *D0L* systems. We also discuss networks of Watson-Crick *D0L* systems.

1 Introduction

Watson-Crick complementarity is a fundamental concept in DNA computing, in a novel challenging scientific area at the interface of theoretical computer science and molecular biology. Although the idea of designing computational devices using DNA as a support for computation has been present for a long time, intensive investigations in this direction started in 1994, after the famous experiment of L. M. Adleman who solved an instance of the Hamiltonian path problem in a laboratory by manipulating DNA strands in a time (counted as the number of operations) being linear in the size of the graph. Since this problem is a so-called intractable computational problem,

*Research supported in part by the Hungarian Scientific Research Fund "OTKA" Grant. No. T 029615

that is, by using computers, the decision about graphs of relatively small size may require an impractical amount of time, the interest of the scientific community immediately turned to the applicability (both theoretical and practical) of DNA as a support for computing. Main questions are, whether or not, properties of DNA strands, their structure and their behaviour, provide possibilities for solving problems in general, which are among these unconventional ways of computing more effective than the known ones, and which give help in approaching problems that cannot be handled by the recent tools.

To answer these questions, mathematical models of DNA should be built and studied, both concerning the structure of the strands and their behaviour. It appeared soon, that using some appropriate formalism based on operations motivated by the recombinant behaviour and properties of DNA strands, universal computational tools can be defined.

The effectivity of computing by using DNA, roughly speaking, originates from two properties, namely, from the massive parallelism provided by the DNA strands and from the phenomenon of Watson-Crick complementarity. The first is due to the fact that DNA strands in a test tube are able to be recombined in parallel, thus, massively parallel operations can be performed. The second feature, the Watson-Crick complementarity, is a phenomenon which is provided by nature "for free", that is, we can be sure without any check that the property holds. In the following we briefly describe main characteristics of this phenomenon.

2 Watson-Crick complementarity

DNA (deoxyribonucleic acid) is found in any living organism as the storage medium for genetic information. It consists of polymer chains, usually called *DNA strands*. A DNA strand is composed of nucleotides; we can also refer them as bases. (From chemical point of view, the two notions are different, but this difference is irrelevant from the point of view of the mathematical theory.) The four nucleotides are A (adenine), G (guanine), C (cytosine), and T (thymine). Adenine and guanine are called *purines*, cytosine and thymine are called *pyrimidines*.

Using some simplification, which does not affect the mathematical theory, throughout we consider DNA strands as strings over the four letter alphabet $\{A, C, T, G\}$.

In nature, the strands form the well-known double helix by bonding of

two separate strands. Bonding of the two strands always takes place by pairwise attraction of the nucleotides : A bonds with T and C bonds with G. This phenomenon is called *Watson-Crick complementarity*, and the pairs (A,T) and (C,G) are said to be *complementary pairs*. Watson-Crick complementarity is provided by nature "for free": whenever in a double strand a nucleotide is found at some position in one of the strands, then we must be sure that in the other strand at the same position the complementary of this nucleotide can be found, no further check of this property is necessary.

Thus, if we have a single DNA strand $ACGGTAC$, then it bonds with the strand $TGCCATG$, and they form the double strand

$ACGGTAC$
$TGCCATG$

This interesting feature was generalized and interpreted as a computational paradigm, called the paradigm of complementarity in [16], [13] as follows: Suppose that the alphabet of the strings to be computed is DNA-like, that is, a complementarity relation is present among the letters. (We shall define DNA-like alphabets and complementarity relations in a more precise manner in the sequel). Then, the following two variants of complementarity paradigms can be formulated:

- The complementarity of two strings leads to some phenomenon such as bonding. Conversely, the occurrence of this phenomenon guarantees that the strings involved indeed are complementary.

- A string induces the complementary string, either randomly or guided by a control device.

The first variant, (i), that is, when the occurrence of a phenomenon implies the complementarity of the involved strings, leads to the well-known twin-shuffle language, and thus, results in the universality of many models of DNA computing.

The second variant, (ii), considers complementarity as an operation: when some conditions on the computed string hold, then the computation continues from the complementary string which is always available.

In the following we recall two types of important models defined according to the above two variants of Watson-Crick complementarity and briefly discuss their remarkable properties. For further models and information the reader is referred to [13] and the references thereof. Before turning to the notions and results, we fix some basic notions and notations.

Throughout we assume that the reader is familiar with the basics of formal language theory. For further details and unexplained notions consult [6], [14], and [13].

The set of nonempty words over an alphabet Σ is denoted by Σ^+; if the empty string, λ is included, then we use notation Σ^*. A set of strings $L \subseteq \Sigma^*$ is said to be a language over Σ. For a string $w \in L$ we denote the length of w by $|w|$, and for a set of symbols U we denote by $|w|_U$ the number of occurrences of letters of U in w.

A morphism $h : V^* \to U^*$ is called a weak coding if $h(a) \in U \cup \{\lambda\}$ for each $a \in V$. If $h : (V_1 \cup V_2)^* \to V_2^*$ is a morphism defined by $h(a) = a$ for $a \in V_1$, and $h(a) = \lambda$ otherwise, then h is called a projection.

3 Universality and Watson-Crick complementarity

The phenomenon of Watson-Crick complementarity provides us with an important reason for using it as a motivation to construct computational paradigms, namely, it guarantees the universality of computation through representing the so-called twin-shuffle language. This property was observed first in [15], and then discussed in details in [16] and [13]. The interested reader can find further information on the topic in [13].

To give some insight into this property, we briefly recall the notion of a twin-shuffle language, following [13]. In its basic form, it is a language over a four-letter alphabet $\Sigma = \{1, 0, \bar{1}, \bar{0}\}$. Let us call letters $1, \bar{1}$, and $0, \bar{0}$ complementary pairs, and let us call for any word $w \in \Sigma^*$, $\bar{w} = h_w(w)$ the complementary word of w, where h_w is an endomorphism of Σ^* into Σ^* ordering to each letter its complementary letter. Then, the shuffle of words w and $\bar{w}$, denoted by $w \sqcup\!\sqcup \bar{w}$, is the set of words $\{x_1 y_1 \dots x_n y_n | w = x_1 \dots x_n, \bar{w} = y_1 \dots y_n, x_i, y_i \in \Sigma^*, 1 \leq i \leq n\}$.

The twin-shuffle language TS over Σ is

$$TS = \{w \sqcup\!\sqcup h_w(w) | w \in \Sigma^*\}.$$

For example, let $w = 01\bar{0}\bar{0}1$, then $\bar{w} = h_w(w) = \bar{0}\bar{1}00\bar{1}$. Then, $z = 01\bar{0}\bar{1}\bar{0}\bar{0}100\bar{1}$ is a shuffle of w and $\bar{w}$.

If we look at the form of the words in the twin-shuffle language, we immediately observe a connection between TS and the double strands of DNA. Let us replace 0 with A, 1 with G, $\bar{0}$ with T, and $\bar{1}$ with C, then from w and $\bar{w}$ above, we obtain two single strands, $AGTTG$ and $TCAAC$, which

are complementary strands of each other. Then, considering the double strand $\begin{matrix} AGTTG \\ TCAAC \end{matrix}$ they form and reading through both strands by turns in arbitrary manner, we obtain strings which belong to TS. It can be easily seen that this method can be applied to any double stranded DNA, thus, any string from TS can be obtained in this manner.

This feature is of particular interest, since it is known [3] that any recursively enumerable language L_0 can be presented as $L_0 = f(TS)$, where f is a generalized sequential mapping and TS is the twin-shuffle language. (For the notion of a generalized sequential mapping see [13], pp. 88-89.)

Thus, any recursively enumerable language can be obtained from the twin-shuffle language (given "for free" by the structure of the double stranded DNA, by the Watson-Crick complementarity) by using some appropriate mapping. The interested reader can find further details in [13] and the corresponding references thereof.

In the following we use the notion of the twin-shuffle language in a generalized sense as follows: suppose that we have an alphabet V and its barred variant $\bar{V} = \{\bar{a} | a \in V\}$. The language $TS_V = \{w \sqcup\!\sqcup \bar{w} | w \in V^*\}$ is called the twin-shuffle language over V. (For a word $w \in V^*$, $\bar{w}$ denotes the string obtained by replacing each symbol in w with its barred version.)

4 Watson-Crick Automata

Observing that the phenomenon of Watson-Crick complementarity leads to the twin-shuffle language, several computational models based on data structures resembling DNA molecules were constructed. Watson-Crick automata, introduced in [4], [5], are particularly interesting types of these constructions. The basic difference between these variants of automata and the usual ones is in the underlying data structure, that is, instead of computing strings, these mechanisms compute pairs of strings resembling DNA molecules, that is, pairs of strings which are complementary words of each other, according to some complementarity relation. For detailed information on these accepting devices the reader should consult [4],[5], [9], and [13].

In the following, by [13], we first present the notion of a Watson-Crick finite automaton, a particularly important variant of Watson-Crick automata, and then, without the aim of completeness, we recall some of its significant properties.

We first need some notations. Let V be an alphabet and let $\rho \subseteq V \times V$ be

a symmetric relation over V. (ρ in some sense refers to the complementarity.)

Let us consider elements (x, y) of $V^* \times V^*$. In accordance to the way how DNA molecules are represented, we can write the pair (x, y) in the form $\begin{pmatrix} x \\ y \end{pmatrix}$. The concatenation of two pairs (x, y) and (u, v) can be written as $\begin{pmatrix} xu \\ yv \end{pmatrix}$, and we use notation $\begin{pmatrix} V^* \\ V^* \end{pmatrix}$ instead of $V^* \times V^*$.

We also denote

$$\begin{bmatrix} V \\ V \end{bmatrix}_\rho = \{ \begin{bmatrix} a \\ b \end{bmatrix} | a, b \in V, (a, b) \in \rho \}$$

and

$$WK_\rho(V) = \begin{bmatrix} V \\ V \end{bmatrix}_\rho^*.$$

The set $WK_\rho(V)$ is said to be the *Watson-Crick domain* associated to the alphabet V and complementarity relation ρ. Elements of $WK_\rho(V)$ can be written in the form $\begin{bmatrix} u \\ v \end{bmatrix}$, where $u, v \in V^*$, and they are called *well-formed double stranded sequences* or simply *double stranded sequences.* The upper sequence, u, is called the *upper strand* and the lower sequence, v, is called the *lower strand.*

The reader should notice important properties of well-formed double stranded sequences: the length of the upper strand is equal to the length of the lower strand and the two strings are complementary words according to the relation ρ.

Now we define the notion of a Watson-Crick finite automaton, according to [13].

A *Watson-Crick finite automaton* (a $WFSA$, for short) is a construct $M = (V, \rho, K, s_0, F, \delta)$, where V and K are disjoint alphabets (the input alphabet, V, and the set of states, K,), $\rho \subseteq V \times V$ is a symmetric relation (the complementarity relation), $s_o \in K$ (the initial state), $F \subseteq K$ (the set of final states), and $\delta : K \times \begin{pmatrix} V^* \\ V^* \end{pmatrix} \to 2^K$ is a mapping such that $\delta(s, \begin{pmatrix} x \\ y \end{pmatrix}) \neq \emptyset$ only for finitely many triples $(s, x, y) \in K \times V^* \times V^*$ (the transition mapping).

Analogously to the finite automaton, we can use notation $s \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} s'$ instead of transition $s' \in \delta(s, \begin{pmatrix} x \\ y \end{pmatrix})$.

Transition $s' \in \delta(s, \begin{pmatrix} x \\ y \end{pmatrix})$ can be interpreted as follows: the automaton is in state s and it passes x at the upper strand and y at the lower strand of a double stranded sequence and then enters state s'.

A transition in a Watson-Crick finite automaton is defined as follows: for $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \begin{pmatrix} u \\ v \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \in \begin{pmatrix} V^* \\ V^* \end{pmatrix}$ such that $\begin{bmatrix} x_1 u x_2 \\ y_1 v y_2 \end{bmatrix} \in WK_\rho(V)$ and $s, s' \in K$, we write

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} s \begin{pmatrix} u \\ v \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \Longrightarrow \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} s' \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \text{ iff } s' \in \delta(s, \begin{pmatrix} u \\ v \end{pmatrix}).$$

We denote by $\Longrightarrow^*$ the reflexive and transitive closure of the relation $\Longrightarrow$.

The reader can easily observe that the Watson-Crick finite automaton works with double stranded sequences.

The complementary relation ρ plays determining role in the notion of a Watson-Crick finite automaton which can be seen as follows: the language defined by a $WFSA$ is the set of upper strings that belong to the well-formed double strands read through by the automaton starting from the initial state and ending with a final state.

Formally, the language accepted by a Watson-Crick finite automaton M is

$$L_u(M) = \{w_1 \in V^* | s_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Longrightarrow^* \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} s_f, s_f \in F, w_2 \in V^*, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)\}.$$

(Subscript u refers to "uncontrolled", the other variant, the "control" language will be defined later.)

Clearly, we also can consider the set of strings at the lower strands or the set of the well-formed double strands as the accepted language, too.

As in the case of customary automaton, several natural variants of Watson-Crick finite automata can be distinguished: a Watson-Crick finite

automaton $M = (V, \rho, K, s_0, F, \delta)$ is *stateless*, if $K = F = \{s_0\}$, *all-final* if $K = F$, *simple* if for all $s' \in \delta(s, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix})$ either $x_1 = \lambda$ or $x_2 = \lambda$ holds, and, finally, 1-*limited* if for all $s' \in \delta(s, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix})$ it holds that $|x_1 x_2| = 1$.

We note that the 1-limited $WFSA$ can also be considered as a particular type of the two-tape and two-head finite automaton, and reversely [13]. Namely, the class of languages recognized by two-tape and two-head finite automata is equal to the class of languages accepted by 1-limited Watson-Crick finite automata. Moreover, the equal accepting power of the 1-limited, the simple, and the arbitrary variants of Watson-Crick finite automata is demonstrated (according to the above definition of the language accepted by these devices) [4], [9], [13].

Although the Watson-Crick finite automaton works with a non-trivial data structure, in particular cases it can accept only languages of a very simple form. For example, for any language L that can be accepted by a stateless Watson-Crick finite automaton $L = L^+$ holds; if L is accepted by a 1-limited stateless automaton, then it is of the form $L = V^+$ for some alphabet V. (For details see [13].)

In the following we denote by $AWK(u)$, $NWK(u)$, $FWK(u)$, $SWK(u)$, and $1WK(u)$ the class of languages accepted by Watson-Crick finite automata of type arbitrary (A), stateless (N), all-final (F), simple (S), and 1-limited (1), respectively. The combined variants are denoted by indicating the corresponding letters together: NS, $N1$, FS, $F1$.

The most important relations among these language classes are summarized as follows [13]:

- $FWK(u) \subseteq AWK(u) = SWK(u) = 1WK(u) \subset CS$,
- $NSWK(u) \subseteq REG \subseteq F1WK(u) \subseteq FSWK(u) \subseteq FWK(u)$ and $NSWK(u) \subset F1WK(u)$,
- $N1WK(u) \subseteq NSWK(u) \subset NWK(u) \subset FWK(u)$.

In [4], in addition to the above defined one, one more language is associated to a Watson-Crick finite automaton, taking transitions into account.

For a Watson-Crick finite automaton $M = (V, \rho, K, s_0, F, P)$ let us consider a labeling $e : P \to Lab$ of transition rules in P (viewing transitions as rewriting rules and denoting this set of rules by P) with elements in the finite set of labels Lab. Let us denote for a computation $\sigma : s_0 w \Longrightarrow^* w s_f$,

where $w \in WK_\rho(V)$, s_0 is the initial state, s_f is a final state, the sequence of labels of transition rules used in σ by $e(\sigma)$.

Then, let us define $L_{ctr}(M)$ (the control language of M) as follows:

$L_{ctr}(M) = \{e(\sigma) | \sigma : s_0 w \Longrightarrow^* w s_f, w \in WK_\rho(V), s_f \in F\}$.

Analogously to the notations above, we denote by $AWK(ctr)$, $NWK(ctr)$, $FWK(ctr)$, $SWK(ctr)$, $1WK(ctr)$, $NSWK(ctr)$, $N1WK(ctr)$, $FSWK(ctr)$, and $F1WK(ctr)$ the classes of control languages accepted by the corresponding variants of Watson-Crick finite automata.

The most remarkable properties of Watson-Crick automata are the possibilities of representing recursively enumerable languages through their accepted languages and some mappings. The next property demonstrates the close connection between Watson-Crick finite automata and the twin-shuffle language, namely, the twin-shuffle language over arbitrary alphabet can be obtained as the set of transition sequences of a stateless 1-limited $WFSA$ [13].

Theorem 4.1 *([13]) For every alphabet V, it holds that $TS_V \in N1WK(ctr)$.*

Then, by the result in [3], the following representations of the recursively enumerable language class hold (see for details [4], [9], and [13]):

Theorem 4.2 *Any recursively enumerable language can be obtained as the image of a deterministic gsm mapping of a language in any of the families $XWK(ctr)$, where $X \in \{A, N, F, S, 1, NS, N1, FS, F1\}$.*

Similarly important representation results can be obtained for recursively enumerable languages in terms of mappings and (uncontrolled) languages recognized by different variants of Watson-Crick finite automata (for details consult [4], [9],[13]). Namely,

Theorem 4.3 *Any recursively enumerable language is the weak coding of a language in any of the language classes $XWK(u)$, where $X \in \{A, F, S, 1\}$.*

Moreover, the following statement holds:

Theorem 4.4 *Any recursively enumerable language can be written in the form $L = h(L')$, where $L' \in AWK(u)$ and h is a projection.*

By these results the reader can observe that the computational completeness can easily be obtained with computation of double stranded strings resembling DNA molecules, any recursively enumerable language can be computed by some appropriate Watson-Crick finite automaton and an appropriate coding mechanims.

In addition to the basic variant of Watson-Crick finite automata, several other natural variants were introduced and studied (for details the reader is referred to [13] and the references thereof): the Watson-Crick finite transducer (an output is associated to the $WFSA$ in the same way as an output is associated to a finite automaton to obtain a generalized sequential machine) or the Watson-Crick reverse automata, where the two strands of the Watson-Crick tape are read from the opposite direction. An interesting variant is the Watson-Crick prefix automaton (a Watson-Crick finite automaton augmented with a Watson-Crick memory) where an additional double stranded tape is associated to the WFSA and the transition is modified as follows: being in some state s, the automaton passes through the strings on the first tape prescribed by the transition, then enters to the next state and writes/reads to/from the second tape the two strings prescribed by the transition. The computation starts from the initial state, and ends if the contents of the first tape is read, the automaton is in a final state, and the second tape contains a well-formed double stranded sequence. Similarly to $WFSA$, these models also lead to representations of the recursively enumerable language class.

In the theory of Watson-Crick automata several questions have remained open, several directions are waiting for examinations. An important aspect is the study of the size complexity of these devices, the conciseness of the description of the different language classes in terms of these automata types. Recently, [12] presented several interesting results about the state- and transition complexity of $WFSA$.

5 Watson-Crick complementarity in the operational sense

The *operational aspect* of Watson-Crick complementarity as a paradigm of computation was introduced and proposed for further study in [10]. The idea behind the notion, formulated in [10], [16], [11], can be expressed as follows: in the course of a developmental or a computational process, things can go wrong to such an extent that it is advisable to continue the process

with the complementary string (which is always available). That is, if some conditions on the current string in computation hold (the conditions of a so-called trigger for turning to the complementary satisfy), instead of the string in generation, the computation continues with the complementary string. The availability of the complementary string is a crucial point in this model: it means that whenever a failure emerges, then there always exists a possibility for correction.

The proposed paradigm leads to various interesting questions and aspects of language theory and, for its general formulation, it can be interpreted in other scientific areas like the theory of developmental systems as well.

The two determining components of the above model are the underlying string computational device (the string rewriting mechanism) and the choice of the trigger for turning to the complementary. The triggers are expected to be sound: the complementary string of a "not correct" word (a string satisfying the trigger for turning to the complementary) must be a "correct" word (conditions for turning to the complementary do not hold in this case), but this is not required for "correct" strings - the complementary of a "correct" string can be either "correct" or "not correct".

As underlying string rewriting mechanisms, Lindenmayer systems (L systems, for short), models of developmental systems [14], appear to be especially suitable. These mechanisms derive strings in a totally parallel manner, at every derivation step each symbol in the generated string is rewritten. Motivated by these observations, a model based on L systems and inspired by the Watson-Crick complementarity in the operational sense, called the Watson-Crick $D0L$ system (the $WD0L$ system, for short), was introduced and examined in [10] and studied further in [11], [16], [17], [7]. In the following, we present the basic notions concerning Watson-Crick $D0L$ systems and briefly recall some of their important and interesting properties discussed in the above articles.

6 Watson-Crick $D0L$ systems

A Watson-Crick $D0L$ system is a $D0L$ system over a so-called DNA-like alphabet Σ and a mapping ϕ which defines the trigger for complementarity transition.

The following presentation of the mathematical formulations of the notions can be found in [10], articles [11],[17], and [7] use slightly different notations.

By a *DNA-like alphabet* Σ we mean an alphabet with $2n$ letters, $n \geq 1$, of the form $\Sigma = \{a_1, \ldots, a_n, \bar{a}_1, \ldots, \bar{a}_n\}$. Letters a_i and $\bar{a}_i$, $1 \leq i \leq n$, are said to be *complementary letters*. Observe that this definition of the complementarity is more restrictive than the definition used in Section 4., above.

The letter-to-letter endomorphism h_w of a DNA-like alphabet Σ mapping each letter to its complementary is said to be the *Watson-Crick morphism.*

When the DNA alphabet $\Sigma_{DNA} = \{A, G, C, T\}$ is considered DNA-like, then purines A and G correspond to the non-barred letters ($a_1 = A$ and $a_2 = G$) and pyrimidines T and C correspond to the barred letters ($\bar{a}_1 = T$ and $\bar{a}_2 = C$). In the sequel we use these terms for any DNA-like alphabet: we call the non-barred letters purines and the barred letters pyrimidines.

By a $D0L$ system we mean a triple $H = (\Sigma, g, w_0)$, where Σ is an alphabet, g is an endomorphism defined on Σ^*, and $w \in \Sigma^*$ is the axiom. The word sequence $S(H)$ of H is defined as the sequence of words $w_0, w_1, w_2, \ldots$ where $w_{i+1} = g(w_i)$ for $i \geq 0$. The language $L(H)$ generated by H is the set of words which appear in $S(H)$. The length sequence $lg(H)$ of H is the sequence $|w_i|$, $i \geq 0$. The growth function f of H is a function $f : N \to N$ with $f(n) = |w_n|$, $n \geq 0$, $w_0, w_1, w_2, \ldots$, being the word sequence of H.

(For details concerning $D0L$ systems the reader is referred to [14] and [6].)

A *Watson-Crick* $D0L$ *system* (a *WD0L system*, for short) is a pair $W = (H, \phi)$, where $H = (\Sigma, g, w_0)$ is a $D0L$ system with a DNA-like alphabet Σ, morphism g and axiom $w_0 \in \Sigma^+$. $\phi : \Sigma^* \to \{0, 1\}$ is a mapping such that $\phi(w_0) = \phi(\lambda) = 0$ and for every word $u \in \Sigma^*$ with $\phi(u) = 1$ it holds that $\phi(h_w(u)) = 0$.

H is said to be the underlying $D0L$ system of W and ϕ is the mapping defining the *trigger for turning to the complementary.*

The derivation in the Watson-Crick $D0L$ system is as follows: when the new string is computed by applying the morphism of the $D0L$ system, then it is checked according to mapping ϕ. If the ϕ-value of the obtained string is 0 (the string is a *correct* word), then the derivation continues in the usual $D0L$ manner. If the obtained string is a *not correct* one, that is, its ϕ-value is equal to 1, then the string is changed for its complementary and the derivation continues with this complementary string. The condition $\phi(u) = 1$ is said to be the trigger for complementarity transition. Notice, however, that $\phi(v) = 0$, $v \in \Sigma^*$, does not imply $\phi(h_w(v)) = 1$, that is, the complementary of a correct word can be either correct or not correct.

A $WD0L$ system, defined as above, is said to be with *regular, context-*

free, context-sensitive, etc. trigger if the set of words in Σ which are not correct according to ϕ is a regular, context-free, context-sensitive language, respectively.

The *word sequence* $S(W)$ of a Watson-Crick $D0L$ system W consists of words $w_0, w_1, w_2, \ldots$, where for each $i \geq 0$

$$w_{i+1} = \begin{cases} g(w_i) & if\ \phi(g(w_i)) = 0 \\ h_w(g(w_i)) & if\ \phi(g(w_i)) = 1. \end{cases}$$

We also can say that w_i *directly derives* w_{i+1} in W, $i \geq 0$, and we can use notation $w_i \Longrightarrow w_{i+1}$. The sequence of derivation steps $w_i \Longrightarrow w_{i+1}$, $i \geq 0$, is said to be the *computation* in W.

The *language*, the *length sequence* and the *growth function* of a $WD0L$ system are defined in the same manner as for $D0L$ systems.

Obviously, various mappings ϕ can satisfy the conditions of defining a trigger for complementarity transition. In the case of *standard Watson-Crick* $D0L$ *systems* a word w satisfies the trigger for turning to the complementary, that is, it is not correct, if and only if it has more occurrences of pyrimidines (barred letters) than purines (non-barred letters).

Formally,
let us consider a DNA-like alphabet $\Sigma = \{a_1, \ldots, a_n, \bar{a}_1, \ldots, \bar{a}_n\}$, $n \geq 1$, and let $\Sigma_{PUR} = \{a_1, \ldots, a_n\}$ and $\Sigma_{PYR} = \{\bar{a}_1, \ldots, \bar{a}_n\}$.

Then, we define $\phi : \Sigma^* \rightarrow \{0, 1\}$ as follows: for $w \in \Sigma^*$

$$\phi(w) = \begin{cases} 0 & if\ |w|_{\Sigma_{PUR}} \geq |w|_{\Sigma_{PYR}}\ and \\ 1 & if\ |w|_{\Sigma_{PUR}} < |w|_{\Sigma_{PYR}}. \end{cases}$$

In the following we recall an example from [10], demonstrating remarkable properties of Watson-Crick $D0L$ systems.

Example 1 Let $W = ((\Sigma, g, w_0), \phi)$ be a standard Watson-Crick $D0L$ system, given as follows: Let $\Sigma = \{a_1, a_2, a_3, \bar{a}_1, \bar{a}_2, \bar{a}_3\}$, and let $g(a_1) = a_1$, $g(a_2) = a_2$, $g(a_3) = a_3$, $g(\bar{a}_1) = \bar{a}_1\bar{a}_2$, $g(\bar{a}_2) = \bar{a}_2$, $g(\bar{a}_3) = \bar{a}_3\bar{a}_3\bar{a}_3$. Furthermore, let $w_0 = a_1a_2\bar{a}_3$ be the axiom.

Let us consider the first few steps of the computation, that is, the first few elements of the word sequence of W. Then, the following strings are

obtained:

$$a_1a_2\bar{a}_3 \quad \mathbf{\bar{a}_1\bar{a}_2a_3^3} \quad \bar{a}_1\bar{a}_2^2a_3^3 \quad \mathbf{a_1a_2^3\bar{a}_3^3} \quad \mathbf{\bar{a}_1\bar{a}_2^3a_3^9} \quad \bar{a}_1\bar{a}_2^4a_3^9$$

$$\bar{a}_1\bar{a}_2^5a_3^9 \quad \bar{a}_1\bar{a}_2^6a_3^9 \quad \bar{a}_1\bar{a}_2^7a_3^9 \quad \bar{a}_1\bar{a}_2^8a_3^9 \quad \mathbf{a_1a_2^9\bar{a}_3^9} \quad \mathbf{a_1a_2^9\bar{a}_3^{27}}$$

$$a_1a_2^{10}\bar{a}_3^{27} \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$

(The strings with boldface letters denote elements of $S(W)$ where a turn to the complementary has taken place in the computational process.)

It can be noticed that the length sequence of the system is strictly growing, due to the barred letters:

$3, \mathbf{5}, 6, \mathbf{7}, \mathbf{13}, 14, 15, 16, 17, 18, \mathbf{19}, \mathbf{37}, 38, \ldots, ...$

(Again, the numbers with boldface symbols refer to the lengths of the strings obtained with a turn to the complementary.)

In [10], by analyzing the word sequence of W, it was shown that the growth function f of W can be computed as follows:

$$f(i) = \begin{cases} 3 & if\ i = 0 \\ 1 + 3^n + 3^{n+1} + k, & if\ i = 3^n + n + k \\ & for\ any\ n \geq 0, 0 \leq k \leq 2 \times 3^n - 1, \\ 1 + 2 \times 3^{n+1}, & if\ i = 3^n + n + 2 \times 3^n, \\ & for\ any\ n \geq 0. \end{cases}$$

In [10], [11] it is proved that the length sequence of W is not a Z-rational sequence. (A sequence $z(i)$ is said to be Z-rational if there is a square matrix M with integer entries such that for every i, $i \geq 1$, $z(i)$ equals the number in the upper right-hand corner of M^i. For details the reader should consult [8].) Briefly, the property is due to the fact that the growth fluctuates between linear growth and exponential growth.

Continuing the examination of the above $WD0L$ system, in [17] it was shown that in the course the computation of words of W a turn to the complementary takes place at the first derivation step and at any derivation step $3^{i+1} + i$ and $3^{i+1} + i + 1$, where $i \geq 0$.

The construction used in this example can be generalized: for example, any $WD0L$ system with the same parameters as W but with a rule of the form $\bar{a}_3 \rightarrow \bar{a}_3^n$, $n > 3$, instead of $\bar{a}_3 \rightarrow \bar{a}_3^3$ demonstrates similar properties. Namely, it determines a not Z-rational length sequence and a computation where the turns to the complementary take place in a not ultimately periodic manner. The reader can easily find more similar examples.

Watson-Crick $D0L$ system raise a lot of interesting questions to study. The reader can immediately notice that any $D0L$ system can be considered as a $WD0L$ system, and $WD0L$ systems are regulated $DT0L$ systems with two tables and special conditions for changing the table. The first obvious, but intriguing problem is, whether the behaviour of $WD0L$ systems differs from the behaviour of ordinary $D0L$ systems ($DT0L$ systems), and if this is the case, what are the differences between the two types of generative mechanisms. Using the example above, [10], and [11] give an answer to this question.

Theorem 6.1 *([10],[11])*
The class of growth functions of Watson-Crick $D0L$ systems contains functions which are not Z-rational.

Since the growth function of any $D0L$ system is Z-rational, the result proves that $WD0L$ systems and $D0L$ systems demonstrate significantly different behaviour. However, there are particular variants of $WD0L$ systems which have $D0L$ growth functions. By a recent result,

Theorem 6.2 *([7]) The growth functions of Watson-Crick $D0L$ systems with regular triggers are $D0L$ growth functions.*

Another statement, proving the essential difference between ordinary $D0L$ systems and Watson-Crick $D0L$ systems can be found in [11]. It is a well-known fact that the alphabets of the words in the word sequence of any $D0L$ system form an ultimately periodic sequence (see for details [14]).

Theorem 6.3 *([11]) The alphabets of the words in the word sequence of a Watson-Crick $D0L$ system do not necessarily form an ultimately periodic sequence, and neither do the prefixes or suffixes of any chosen length.*

A basic determining feature of $WD0L$ systems is the possibility to turn to the complementary under computing. A problem defined by this property is the problem of the *stability* of the system, that is, to decide whether or not in the course of the computation a turn to the complementary will take place. A Watson-Crick $D0L$ system $W = ((\Sigma, g, w_0), \phi)$ is said to be *stable* if the complementarity transition never takes place in the sequence $S(W)$, that is, the sequence consists of the words $g^i(w_0)$, $i \geq 0$.

In [17] it is shown that

Theorem 6.4 *([17]) The stability problem is decidable for Watson-Crick D0L systems with regular triggers but undecidable for systems with context-sensitive triggers.*

In addition, in [11] in the case of standard Watson-Crick $D0L$ systems this problem has been shown to be equivalent to a famous problem with open status, called Z_{pos}, which is formulated as follows: Given a Z-rational sequence $z(i)$, decide whether or not $z(i) \geq 0$ holds for all $i \geq 0$. (For further details the reader is referred to [18] and [8].

Theorem 6.5 *([11]) Any algorithm for solving the stability of a given standard Watson-Crick D0L system can be converted to an algorithm for solving the problem Z_{pos}, and conversely.*

An important notion concerning Watson-Crick $D0L$ systems is the *Watson-Crick road*, introduced and discussed in details in [17].

Let $W = (H, \phi)$ be a Watson-Crick $D0L$ system, where $H = (\Sigma, g, w_0)$. The *Watson-Crick road* of W is an infinite binary word α over $\{0, 1\}$ such that the i-th bit of α is equal to 1 if and only if at the i-th step of the computation in W a transition to the complementary takes place, otherwise it is 0. That is, $\phi(g(w_{i-1})) = 1$, where w_i, $i \geq 1$, is the i-th element of the word sequence $S(W)$ of W. Notice, while stability refers to the existence of a turn to the complementary in the course of the computation, the Watson-Crick road registers every derivation step with this property.

In [17], among other things, it was shown that

Theorem 6.6 *([17]) The Watson-Crick road of a standard Watson-Crick D0L system is not necessarily ultimately periodic.*

As for other language generating mechanisms, *equivalence questions* are important problems for Watson-Crick $D0L$ systems, too. Two Watson-Crick $D0L$ systems, W_1 and W_2, are said to be *sequence equivalent* if $S(W_1) = S(W_2)$ holds, *language equivalent* if $L(W_1) = L(W_2)$ satisfies ($L(W_i)$, $i = 1, 2$ is the language of W_i), and *growth equivalent*, if for their growth functions f_1 and f_2, respectively, $f_1(i) = f_2(i)$ holds for each i, $i \geq 0$. Moreover, W_1 and W_2 are said to be *road equivalent*, if their Watson-Crick roads coincide.

In [17] and in [7] the following statements can be found:

Theorem 6.7 *([17],[7]) The sequence, language, growth, and road equivalence problems are all undecidable for Watson-Crick D0L systems with*

context-sensitive triggers, but decidable for Watson-Crick D0L systems with regular triggers.

In the case of standard system, from algorithmic point of view, these problems can be converted to Z_{pos}.

Theorem 6.8 *([17]) Any algorithm for solving the sequence, language, and growth equivalence problem for standard Watson-Crick D0L systems can be converted into an algorithm for solving the problem Z_{pos}.*

7 Networks of Watson-Crick $D0L$ systems

Recent developments in the investigations of Watson-Crick $D0L$ systems are the examinations concerning their behaviour in a distributed architecture. For this purpose, so-called networks of Watson-Crick $D0L$ systems were introduced and offered for further study in [2].

A *network of Watson-Crick D0L systems* (an $NWD0L$ system, for short), is a finite set of Watson-Crick $D0L$ systems over the same DNA-like alphabet and with the same trigger (nodes or components of the network) which $WD0L$ systems act on their own strings in a synchronized manner and after each derivation step communicate some of the obtained words to each other. The condition for communication is determined by the trigger for complementarity transition. In [2] two variants of communication protocols were introduced: in the case of protocol (a), after performing a derivation step, the node keeps every obtained correct word and the complementary of each obtained not correct word (each corrected word) and sends a copy of every corrected word to each other node. In the case of protocol (b), as in the previous case, the node keeps all the correct words and the corrected ones (the complementaries of the not correct strings) but communicates a copy of every correct string to each other node. The two protocols realize different communication strategies: in the first case, if some error is detected, it is corrected and a note is sent about this fact to the others. In the second case, the nodes inform the other nodes about their correct activities and keep all information which refers to correction of some error.

The concept raises several interesting problems to study: comparison of the behaviour of the networks with different communication strategies, description of the dynamics of the string collections at the nodes, properties of computation in terms of these devices. In the following, by [2] and [1],

we briefly recall the basic notions and some remarkable properties of these systems.

By an N_rWD0L *system* (a *network of Watson-Crick D0L systems with r components or nodes*), where $r \geq 1$, we mean an $r+2$-tuple $\Gamma = (\Sigma, \phi, (g_1, \{A_1\}),$
$\ldots, (g_r, \{A_r\}))$, where $\Sigma = \{a_1, \ldots, a_n, \bar{a}_1, \ldots, \bar{a}_n\}$ is a DNA-like alphabet, the alphabet of the system, $\phi : \Sigma^* \to \{0,1\}$ is a mapping defining a trigger for complementarity transition, and (g_i, A_i), $1 \leq i \leq r$, called the i-th component or the i-th node of Γ, is a pair where g_i is a $D0L$ morphism over Σ and A_i is a correct nonempty word over Σ, called the axiom of the ith component.

If the number of the components is irrelevant, then we speak of an $NWD0L$ system.

The $NWD0L$ system Γ is said to be *standard*, if the Watson-Crick $D0L$ systems defined by its components are standard systems. (Recall that in this case ϕ defines any word with more occurrences of barred letters than non-barred letters to be a not correct word, the words with more occurrences of non-barred letters than barred ones are considered as correct words.)

An instantaneous description of the $NWD0L$ system is given by its states.

For an N_rWD0L system $\Gamma = (\Sigma, \phi, (g_1, \{A_1\}), \ldots, (g_r, \{A_r\}))$, $r \geq 1$, the r-tuple $(L_1, \ldots, L_r)$, where L_i, $1 \leq i \leq r$, is a finite set of correct strings over Σ, is called a *state* of Γ. L_i, $1 \leq i \leq r$, is called the state of the i-th component. $(\{A_1\}, \ldots, \{A_r\})$ is said to be the initial state of Γ.

$NWD0L$ systems change their states by *direct derivation steps.* A direct change of a state to another one means a rewriting step followed by (possibly empty) communication according to the given protocol of the system.

Let $s_1 = (L_1, \ldots, L_r)$ and $s_2 = (L'_1, \ldots, L'_r)$ be two states of an N_rWD0L system $\Gamma = (\Sigma, \phi, (g_1, \{A_1\}), \ldots, (g_r, \{A_r\}))$, $r \geq 1$. We say that

- s_1 directly derives s_2 by protocol (a), written as $s_1 \Longrightarrow_a s_2$, if $L'_i = C'_i \cup_{j=1}^{r} h_w(B'_j)$, where $C'_i = \{g_i(v) | v \in L_i, \phi(g_i(v)) = 0\}$ and $B'_j = \{g_j(u) | u \in L_j, \phi(g_j(u)) = 1\}$, $1 \leq i, j \leq r$, and

- s_1 directly derives s_2 by protocol (b), written as $s_1 \Longrightarrow_b s_2$, if $L'_i = h_w(B'_i) \cup_{j=1}^{r} C'_j$, where $B'_i = \{g_i(u) | u \in L_i, \phi(g_i(u)) = 1\}$ and $C'_j = \{g_j(v) | v \in L_j, \phi(g_j(v)) = 0\}$, where $1 \leq i, j \leq r$.

We denote by $\Longrightarrow_x^*$ the reflexive and transitive closure of $\Longrightarrow_x$, where $x \in \{a, b\}$.

Thus, in the case of both protocols, after applying a derivation step in the $WD0L$ manner, the node keeps the correct words and the corrected words (the complementaries of the not correct ones), and in the case of protocol (a) it sends a copy of every corrected word to each other node, while in the case of protocol (b) it communicates a copy of every correct word to each other node.

The *state sequence* of an N_rWD0L system Γ, defined above, is $S(\Gamma) = s(0), s(1), \ldots$, where $s(0) = (\{A_1\}, \ldots, \{A_r\})$ and $s(t) \Longrightarrow_x s(t+1)$ for $t \geq 0$, $(x) \in \{a, b\}$.

The sequence of direct derivation steps in Γ is the *computation* of Γ.

By the *language* of Γ we mean

$$L(\Gamma) = \{u_1 \in L_1 \mid (\{A_1\}, \ldots, \{A_r\}) \Longrightarrow_x^* (L_1, \ldots, L_r)\},$$

that is, the set of words which occur at the first node at some step of the computation.

It is an important question whether or not there are different networks of Watson-Crick $D0L$ systems which use different communication protocols but determine the same or almost the same dynamics of sets of strings, that is, whether a state sequence of a network using a certain protocol can be obtained by another network using another protocol variant. In [2] a partial answer is given to this question. Namely, it is shown that there are restricted variants of $NWD0L$ systems (called s-type $NWD0L$ systems) which working with protocol (a) determine the same state sequences as $NWD0L$ systems of the same restricted type with protocol (b), and reversely, supposing that they are defined over the same alphabet and they have the same trigger for complementarity transition. A $NWD0L$ system W is said to be *s-type* if in the course of the computation any correct word occurring at some node of W (being an element of the state of a node) has a not correct complementary word. In this case the two protocols can be considered as "duals" of each other.

Theorem 7.1 *([2]) For every s-type N_rWD0L system Γ with protocol (x), there exists an s-type N_rWD0L system Γ' with protocol (y), where $x \neq y$, $x, y \in \{a, b\}$, such that Γ and Γ' have the same state sequences.*

Networks of Watson-Crick $D0L$ systems determine string collections changing in time. One measure describing the dynamics of these collections is the number of strings present in the network (at some nodes, at a specific node) at a certain step of the computation.

Let $\Gamma = (\Sigma, \phi, (g_1, \{A_1\}), \ldots, (g_r, \{A_r\}))$, $r \geq 1$, be an N_rWD0L system with protocol (x), $x \in \{a, b\}$, and let $s(t) = (L_1(t), \ldots, L_r(t))$, $t = 0, 1, \ldots$, be the state of Γ at step t of the computation. Then function $p : N \to N$ defined by $p(t) = \sum_{i=1}^{r} card(L_i(t))$, $t \geq 0$, is called the *string population growth function* of Γ.

String population growth functions of $NWD0L$ systems exhibit remarkable properties.

Theorem 7.2 *([2]) For $x \in \{a, b\}$ there exists an $NWD0L$ system Γ working with protocol (x) such that the string population growth function of Γ is not a Z-rational function.*

The statement is proved by demonstrating an example for an $NWD0L$ system with a not Z-rational string population growth function, constructed by using analogous ideas to that are used in the construction of the $WD0L$ system, cited as Example 1, above.

For protocol (a), the $NWD0L$ system is a standard $NWD0L$ system $\Gamma = (\Sigma, \phi, (g_1, a_1), (g_2, a_1a_2\bar{a}_3))$, where $\Sigma = \{a_1, a_2, a_3, \bar{a}_1, \bar{a}_2, \bar{a}_3\}$, $g_1(b) = a_1$ for $b \in \Sigma$, $g_2(a_i) = a_i$, $1 \leq i \leq 3$, $g_2(\bar{a}_1) = \bar{a}_1\bar{a}_2$, $g_2(\bar{a}_2) = \bar{a}_2$ and $g_2(\bar{a}_3) = \bar{a}_3^3$. Then, it can be proven that the first node in this network is a black hole (a node which never emits any string) and the second node communicates a new string to the first one in a not ultimately periodic manner, which implies that the function taking values that are equal to the differences between the number of strings in the system in two subsequent states cannot be a Z-rational function, and thus, the string population growth function of Γ cannot be Z-rational either. For protocol (b) the statement follows by simple modifications of the definition of Γ above.

Communication in networks of Watson-Crick $D0L$ systems raises a lot of intriguing questions. Among them a particularly interesting problem is whether or not a given network contains a *black hole*, that is, a node which never emits any string in the course of the computation. Let $\Gamma = (\Sigma, \phi, (g_1, \{A_1\}), \ldots, (g_r, \{A_r\}))$, $r \geq 1$, be an N_rWD0L system with protocol (x), $x \in \{a, b\}$ and let $s(t) = (L_1(t), \ldots . L_r(t))$, be the state of Γ at derivation step t, where $t \geq 0$. We say that the ith component of the system is a black hole, $1 \leq i \leq r$, if for every $t \geq 0$, $g_i(L_i(t))$ consists of

correct words for protocol (a), and $g_i(L_i(t))$ consists of not correct words for protocol (b).

The problem of the existence of a black hole in the network is in close connection with the stability of the strings occurring at the nodes under computation in the $NWD0L$ system. Since an $NWD0L$ system can consists of exactly one component, the following simple observation can be made:

Theorem 7.3 *([2]) Any algorithm for deciding whether a standard $NWD0L$ system working with protocol (a) contains a black hole can be converted to an algorithm for solving problem Z_{pos}.*

$NWD0L$ systems determine strings under computation, thus, they can be considered as language determining devices as well. The following result from [1] demonstrates that any recursively enumerable language can be represented in terms of languages computed by standard networks of Watson-Crick $D0L$ systems.

Theorem 7.4 *([1]) For each recursively enumerable language L over an alphabet V there exists a standard $NWD0L$ system Γ such that $L = L(\Gamma) \cap V^*$ holds.*

8 Concluding remarks

The models briefly discussed in the paper form important computational paradigms motivated by the Watson-Crick complementarity. In the future, both the investigations of their properties and the study of their potential applicability in other scientific areas would be of interest.

References

[1] E. Csuhaj-Varjú: Computing by networks of standard Watson-Crick $D0L$ systems. In: Proc. Workshop on Algebraic Systems, Formal Languages and Computations, Kyoto, 21-23 March, 2000. (M. Ito, ed.), to appear.

[2] E. Csuhaj-Varjú, A. Salomaa: Networks of Watson-Crick $D0L$ systems. Presented at 3rd Int. Colloquium on Words, Languages and Combinatorics, March 14-18, 2000, Kyoto. Submitted to the proceedings.

[3] J. Engelfriet, G. Rozenberg: Fixed point languages, equality languages, and representations of recursively enumerable languages. Journal of the ACM 27 (1980), 499-518.

[4] R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa: Watson-Crick finite automata. Technical Report 97-13, Dept. of Computer Science, Leiden University, 1997.

[5] R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa: Watson-Crick finite automata. Proc. of the Third Annual DIMACS Symposium on DNA Based Computers, Philadelphia, 1997, 305-317.

[6] Handbook of Formal Languages. Vol. I-III. (G. Rozenberg, A. Salomaa, eds.) Springer Verlag, Berlin-Heidelberg-New York, 1997.

[7] J. Honkala, A. Salomaa: Watson-Crick *D0L* systems with regular triggers. TUCS Technical Report 360, August, 2000.

[8] W. Kuich, A. Salomaa: Semirings, Automata, Languages. EATCS Monographs on Theoretical Computer Science, Springer Verlag, Berlin-Heidelberg-New York, 1986.

[9] C. Martin-Vide, Gh. Păun, G. Rozenberg, A. Salomaa: Universality results for finite H systems and Watson-Crick finite automata. In: Computing with Bio-Molecules. Theory and Experiments (Gh. Păun, ed.), Springer, Berlin, 1998, 200-220.

[10] V. Mihalache, A. Salomaa: Watson-Crick *D0L* systems. EATCS Bulletin 62 (1997), 160-175.

[11] V. Mihalache, A. Salomaa: Language Theoretic Aspects of DNA Complementarity. TUCS Technical Report 202, September, 1998. Also: Theoretical Computer Science, to appear.

[12] A. Păun, M. Păun: State and transition complexity of Watson-Crick Finite Automata. In: Proc. FCT'99 (G. Ciobanu, Gh. Păun, eds.), LNCS 1684, Springer, Berlin, 1999, 409-420.

[13] Gh. Păun, G. Rozenberg, A. Salomaa: DNA Computing - New Computing Paradigms. EATCS Text Series, Springer, 1998.

[14] G. Rozenberg, A. Salomaa: The Mathematical Theory of L systems. Academic Press, New York, London, 1980.

[15] G. Rozenberg, A. Salomaa: Watson-Crick complementarity, universal computations and genetic engineering. Technical Report 96-28, Dept. of Computer Science, Leiden, 1996.

[16] A. Salomaa: Turing, Watson-Crick and Lindenmayer. Aspects of DNA Complementarity. In: Unconventional Models of Computation. (C.S. Calude, J. Casti, M.J. Dinneen, eds.) Springer Verlag, Singapore-Berlin-Heidelberg-New York, 1998, 94-107.

[17] A. Salomaa: Watson-Crick Walks and Roads on $D0L$ Graphs. Acta Cybernetica 14 (1) (1999), 179-192.

[18] A. Salomaa, M. Soittola: Automata-Theoretic Aspects of Formal Power Series. Text and Monographs in Computer Science. Springer Verlag, Berlin-Heidelberg-New York, 1978.

Computing with Biomolecules

Nataša Jonoska
Department of Mathematics
University of South Florida
Tampa, Florida 33620

Abstract

This article gives a brief description of DNA structure and basic enzyme operations that have been used in biomolecular computing. It describes the first successful experiment in DNA computing, the theoretical model of simple splicing systems and the possibility of using three dimensional DNA structures in DNA based computations.

1 Introduction

Ever since the seminal Adleman's paper in 1994 [1] DNA computing, as is now known as a separate field, energized the scientific minds of many molecular biologists, computer scientists, chemists, mathematicians... In [1], Adleman solved a small instance of the well known Hamiltonian Path Problem solely by using DNA molecules and techniques from molecular biology. Although the instance solved was tiny, the potential for using DNA molecules for computing opened a field for new ideas in many different sciences. The observation that the complexity of each living being is a result of several simple biological processes (transcription, replication, recombination...) that are guided by the information that is encoded in DNA, reminded many mathematicians to a well known fact in recursion theory. Every computable function can be obtained by operations applied to a basic set of functions using a small number of simple rules [9]. This observation accounts for the appearance of many theoretical models of DNA computers, capable of performing the complex computations of a universal Turing machine (for example [2, 15, 20, 32, 33, 35, 49]).

Though at this time many theoretical models have been proposed, the experimental verification of their feasibility is yet to be determined. Only very small instances of computational problems have been solved experimentally ([1, 30, 29, 11]). But the investigations and the experiments in progress at several institutions worldwide are scientifically significant. They bring together scientists with deep knowledge of many different fields that it is most certain that their contributions will elevate both science and technology.

This paper is meant to be a short introduction of the basic ideas in DNA computing for readers that have background in computer science and might not be familiar with the basic DNA structure. In particular we give a short description of DNA structure and some of the operations with enzymes that have been used in several models of DNA computers. The topics covered within this article reflect the interest of the author and the reader is advised to check the bibliography at the site given in [12] for many other ideas. The paper is organized in the following way. Section 2 gives a short description (needed for the purpose of this article) of the structure of DNA molecule and in vitro enzyme manipulation of DNA. In Section 3 we concentrate on DNA computing models. We start with explanation of the Adleman's experiment. The basic idea of the theoretical model of splicing (H-) systems is introduced through the model of simple splicing systems. We end with another theoretical idea of using three dimensional DNA structures in computing.

2 The Structure of DNA and Operations Aided by Enzymes

2.1 The structure of DNA

A more thorough and not very technical description of the structure of DNA and the operations performed by enzymes can be found in [16, 31] (see also [20]). Detailed laboratory protocols can be found in [4].

The basic element in the very well known double helix of a DNA molecule is a DeoxyriboNucleic Acid (which is also known as *nucleotide* for short). Nucleotides have three parts, phosphate group (through which nucleotides are connected in a string), sugar, and nitrogenous base. The sugar contains five carbons that are enumerated $1'$ through $5'$ (not to be confused with the carbons in the base enumerated 1 through 5). The $5'$ carbon is connected to the phosphate group, the $1'$ carbon is connected to the base and the $3'$

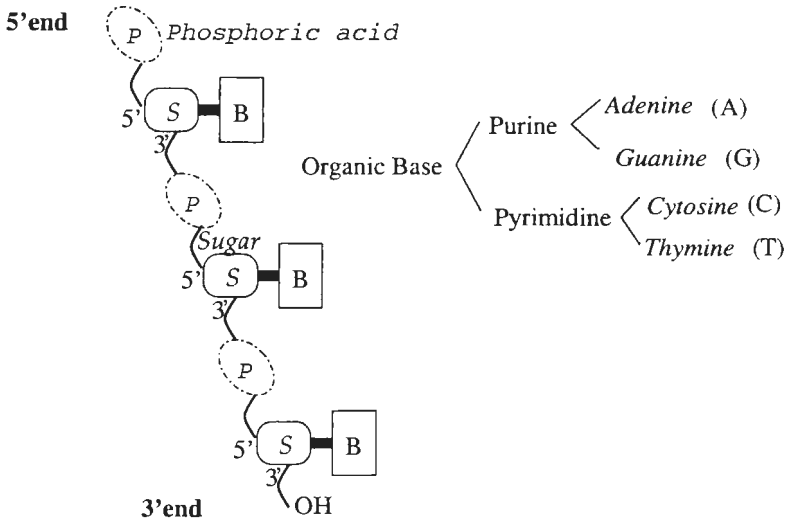


Figure 1: Single strand of DNA

carbon has a hydroxyl group which will be called 3′ end. The phosphate group hanging out of the 5′ end can join with the hydroxyl group at the 3′ end to form a strong (covalent) bond known as phosphodiester bond. This is ilustrated in Figure 1. Through this connection we obtain *one strand* of DNA i.e. a sequence of nucleotides bonded with phosphodiester bonds. At one end of the strand there is a "free" 5′ end and at the other end of the strand there is a "free" 3′ end. We say that the strand is oriented and the conventional notation is $5' \longrightarrow 3'$.

Nucleotides differ only by their bases which come in four different types: *adenine, guanine, cytosine, thymine* which in short are denoted with A,G,C and T. Since nucleotides differ only by their bases, they are also called A, G, C, and T. The bases of two different nucleotides may bond together with a weak bond and form the so called hydrogen bond. There are two ways that these bonds can form: adenine with thymine (A-T paring) and guanine with cytosine (G-C paring). We say that A is complementary to T and G is complementary to C. This is the well known *Watson-Crick complementarity.* Note that the hydrogen bond happens such that the 5′ end of one nucleotide is on the opposite side of the 5′ end of its complementary nucleotide. Hence, the complementarity of A-T and G-C also assumes opposite orientation of

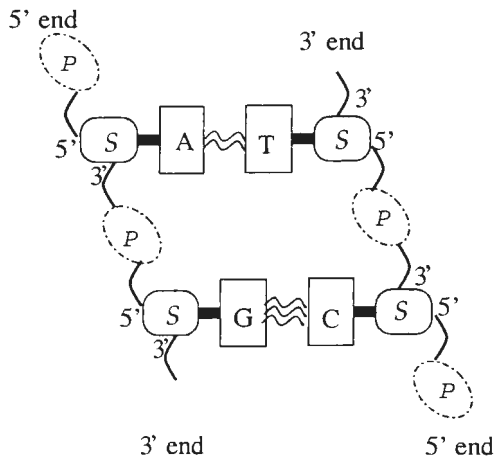


Figure 2: Watson-Crick (WC) pairing

the nucleotides (see Figure 2). Two strands of DNA with complementary sequences of nucleotides (with opposite orientations) join together through hydrogen bonds and form a double stranded DNA. The chemistry of this molecule is such that the phosphodiester bonds of the two strands form a double helix. This is illustrated at Figure 3.

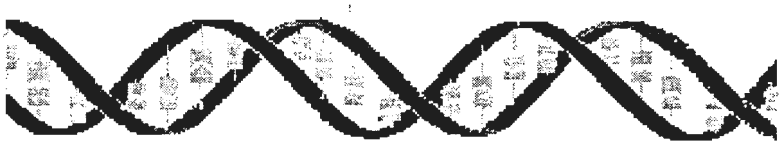


Figure 3: The double helix of a double stranded DNA

2.2 Operations with enzymes

2.2.1 Polymerase.

DNA polymerases are enzymes that synthesize DNA. With these enzymes, a DNA strand can be duplicated or extended. Assume that we have a DNA

strand which we will call *template* and we know the sequence of a short portion of it (say around 20 nucleotides). Chemically, short single stranded DNA sequences can be synthesized which are then called olygonucleotides, or just oligos. Hence we can synthesize the complement of the known portion of the template. We call this oligo a *primer* which forms hydrogen bond (aneals) with the template. The 3′ end of the primer has a free hydroxil group and the polymerase extends this 3′ end by forming phosphodiester bonds with new nucleotides in a way that the new sequence complements the template sequence. This process is schematically represented in Figure 4.

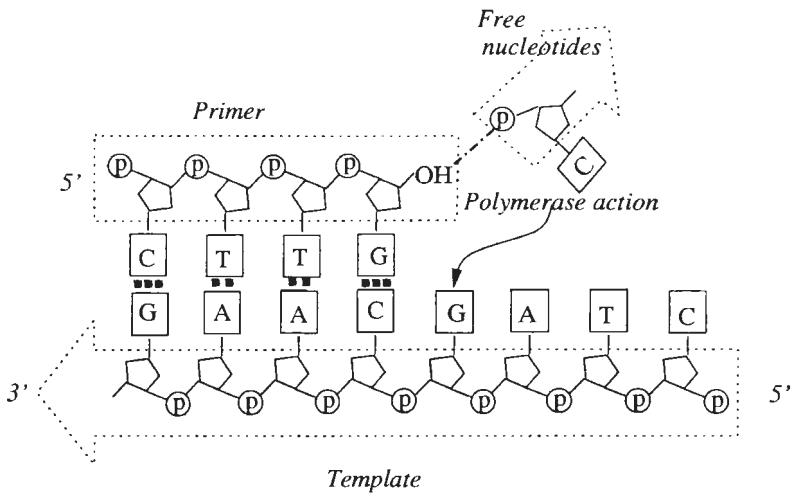


Figure 4: The action of DNA polymerase

One of the most commonly used protocol in molecular biology is the so called *polymerase chain reaction* (PCR). This reaction detects and produces a large number of molecules from an existing pool of molecules. This molecule (template) is copied in such a large number, that the rest of the molecules in the solution are undetectable in comparison. This method is used to detect or extract a certain sequence within a large mix of molecules.

The reaction works in the following way. Two primers are synthesized, one complementary to the 3′ end of one of the strands of the template and the other primer complementary to the 3′ end of the other strand of the template (see Figure 5). A small amount of a complex mix of molecules is

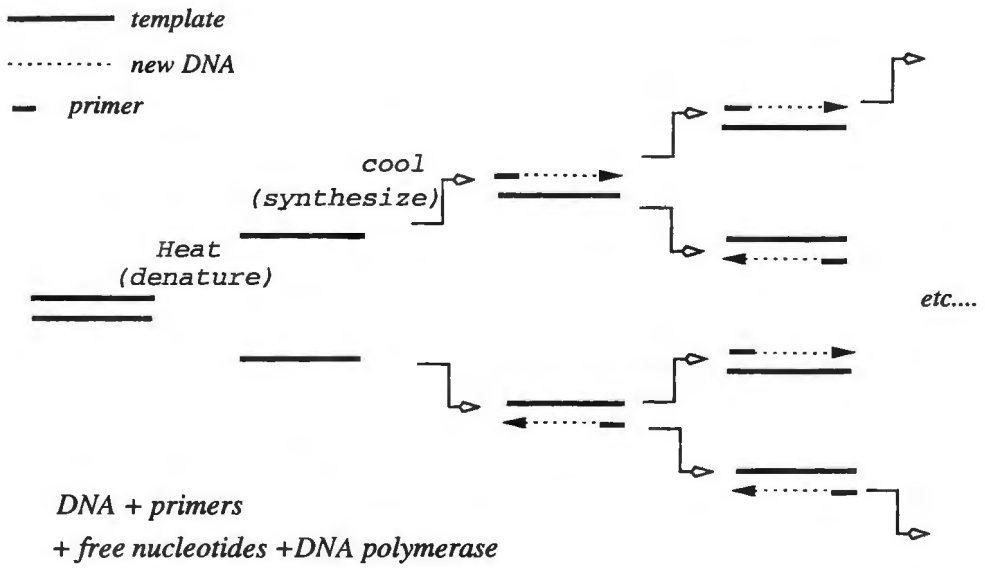


Figure 5: Plymerase Chain Reaction (PCR)

added in a solution. Also, a large amount of primers, free nucleotides of all four types, and a thermostable polymerase enzyme is added. Additional chemicals are also needed, but they are irrelevant for our discussion. The solution is put in a thermocycler that heats and cools at given temperatures. The weak hydrogen bonds are broken when the molecules are heated to 95°C. When molecules are cooled down, the complementary strands reaneal. Since there are much more primers in the solution than there are templates, the primers aneal to the template molecules. The polymerase then extends the primers and duplicates the templates. The heating and cooling cycle is usually repeated about 30 times. Since each strand of the double stranded template molecule can potentially be duplicated in this way, the number of template molecules potentially can increase exponentially with each thermocycle. This reaction is illustrated in Figure 5.

2.2.2 Type II restriction enzymes (endonucleses).

These enzymes recognize specific sequences of nucleotides in a double stranded DNA molecule and cut the molecule by destroying the phosphodiester bonds

at specific places of the two strands. Endonucleases come from different organizms and their names are given according to their origin. Different enzymes recognize different sequenes of nucleotides, and even if they recognize the same sequence, they may cut the molecule in a different way. For example, *Eco*RI recognizes the sequence $\begin{matrix}^{5'}\text{GAATTC}^{3'}\\ {}_{3'}\text{CTTAAG}_{5'}\end{matrix}$ and it cuts a double stranded molecule $\begin{matrix}\ldots\text{N N G A A T T C N N}\ldots\\ \ldots\text{N N C T T A A G N N}\ldots\end{matrix}$ (where "N" stands for any nucleotide) into two pieces:

$$\begin{matrix}\ldots\text{N N G}^{3'}\\ \ldots\text{N N C T T A A}^{5'}\end{matrix} \quad \text{and} \quad \begin{matrix}{}^{5'}\text{A A T T C N N}\ldots\\ {}_{3'}\text{G N N}\ldots\end{matrix}.$$

The single stranded portion AATT is called *overhang* and we say that *Eco*RI leaves a $5'$ overhang. Two other enzymes *Xma*I and *Sma*I recognize the same sequence $^{5'}\text{CCCGGG}^{3'}$, but after cutting, *Xma*I leaves a $5'$ overhang CCGG and *Sma*I cuts separating CCC and GGG without any overhang (such cuts are called "blunt" cuts). There are hundreds known and commercially available such restriction site enzymes.

2.2.3 Ligase

It is said that a double stranded DNA molecule has a *nick* if the phosphodiester bond between two consecutive nucleotides within one of the strands is broken. A ligase is an enzyme that closes the nicks, i.e. recovers the broken bonds. There are two most common ways that this enzyme is used in the models of DNA based computers. The first one is to "glue" two or more single stranded molecules together. This idea was used by Adleman [1] in his experiment. Assume we have two single stranded molecules (call them a and b) and we want to join the $3'$ end of a with the $5'$ end of b. We can synthesize an oligo that, say is 20 nucleotides long, such that the ten nucleotides at the $5'$ end are complementary to the ten nucleotides of the $3'$ end of a and the other ten nucleotides at the $3'$ end are complementary to the the nulceotides at the $5'$ end of b. This is presented in Figure 6 where the synthesized oligos are 12 nucleotides long. In this case the new olygo plays a role of a "linker" that aneals (forms the hydrogen bonds) to a and b. Once this link is formed, the ligase can establish the phosphodiester bond between the $3'$ end of a and $5'$ end of b and a strand ab can be formed.

Another way to use the ligase is in a combination with the restriction site enzymes. Assume that an enzyme has cut two molecules leaving four pieces

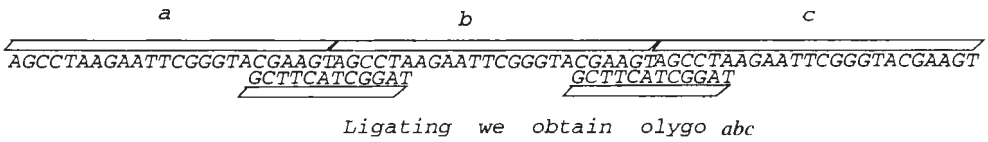


Figure 6: Extending DNA strands with linkers

of DNA each with an overhang. This is illustrated in Figure 7 (a). Then these pieces can recombine (i.e. we can allow for the hydrogen bonds to be established between the complementary parts) and besides the old ones, produce two new molecules Figure 7 (b). A ligase is needed to close the nicks (establish the phosphodiester bonds), and make the molecule stable.

2.2.4 Length Selection

There is a technique called "gel electrophoresis" which separates DNA molecules by their length. The DNA is negatively charged and after being placed in a small well of a gel in an active electric field, it slowly starts to move towards the positive side. Larger molecules move slower and smaller molecules move faster. One way to see double stranded DNA molecules is by staining the gel with ethidium bromide which binds to double stranded DNA and shines under ultraviolet light. Under ultraviolet light one can see different bands showing segments of DNA with different length, shorter being closer to the positive end and longer being closer to the negative end of the electric field. DNA molecules within one band have the same length. Knowing the distance traveled by the molecules, one can compute their length (this is most commonly done by using a commercially available "ladder" which is a set of molecules with known lengths that are placed in a separate well of the gel, in parallel with the molecules of the experiment). The portion of the gel that contains molecules with the desired length can be cut out of the gel, the DNA purified, and then, this DNA can be used in subsequent experiments.

2.2.5 Circular Strings and Junction Molecules.

In nature the DNA in the cells can be linear double stranded molecule (in eukaryotes) but also it can be in a circular form (mostly in viruses and bacteria, prokaryotes). Circular DNA can be obtained also in a laboratory

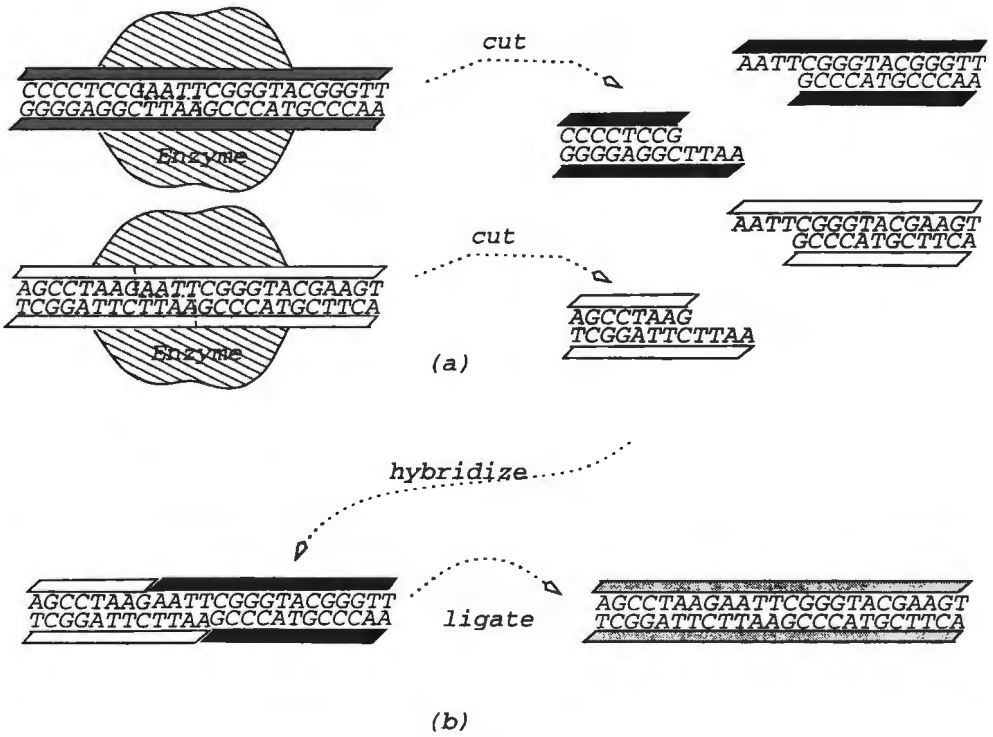


Figure 7: The action of a restriction site endonuclease and a ligase.

by joining (ligating) the ends of a linear DNA. Such molecules are used in several models of DNA based computers (see for ex. [31], Chapter 9).

The WC-complementarity of the nucleotides is one of the leading properties used in DNA based computing. Many authors have used this property to form other, more complicated DNA structures. We are witnessing an explosion in new laboratory techniques and products of biotechnology. Chemistry laboratories are using synthesized DNA for designing and assembling unusual three dimensional DNA structures [40]. Junction molecules, used in the following section, are fairly well understood. These molecules were used to construct DNA polyhedra, a quadrilateral, a truncated octahedron and a cube [6, 7, 44]. Knots have been shown to be rather easy constructions (compared to other structures). In theory, virtually any knot can be constructed using right-handed B-DNA for negative crossings and left-handed

Z-DNA for positive crossings [43, 40, 44, 10]. Many catenanes and linkages of DNA molecules are known, but just recently, using B and Z DNA, the first Borromean DNA rings were assembled [28]. Another very important step towards constructing three dimensional DNA crystals was made by the design and the assembly of a two dimensional lattice made of DNA tiles [45, 50].

The k-armed branched molecules (k is a natural number ≥ 2) seem to be suitable for graph construction. An example of 4-armed branched molecule is presented in figure Fig. 8. In this figure, the double helix of the molecule is not presented. Hydrogen bonds between the anti-parallel, complementary Watson-Crick (WC) bonds are depicted as dotted segments between the strands. Polarity of the DNA strands is indicated with arrowheads being placed at the 3′end. The angles between the "arms" are known to be flexible. If we allow each "arm" to be 200 to 300 base pairs long, then the "arms" of this molecule become rather flexible also and we deliberately show them curved. The 3′ends are extended such that each arm ends with a single strand. This single stranded part will join its WC complement once placed in a test tube.

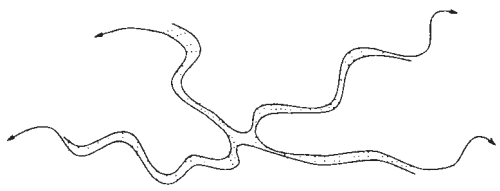


Figure 8: A four armed branched junction molecule.

Construction and properties of these molecules are fairly well understood [41, 42]. By making the body of the connection between the arms larger, it seems that we do not have to put any restrictions on how many arms a molecule can have [39]. In Section 3.3 we will see how 3D structures, in particular junction molecules can be used to solve the three vertex colorability problem.

3 Some models for DNA based computers

3.1 The Adleman's experiment

Adleman solved a special case of the Hamiltonian Path Problem for a directed graph G by generating paths using DNA molecules. This graph is depicted in Fig. 9. Here we give a short description of his experiment.

The Hamiltonian Path Problem (HPP) asks whether in a given (directed) graph G there is a path from one vertex (denote it with v_{in}) to another vertex (denote it with v_{out}) that visits every other vertex exactly once. If such path exists then it is called Hamiltonian. For the example in Figure 9, v_{in} is the vertex 0 and v_{out} is the vertex 6. Given a directed graph

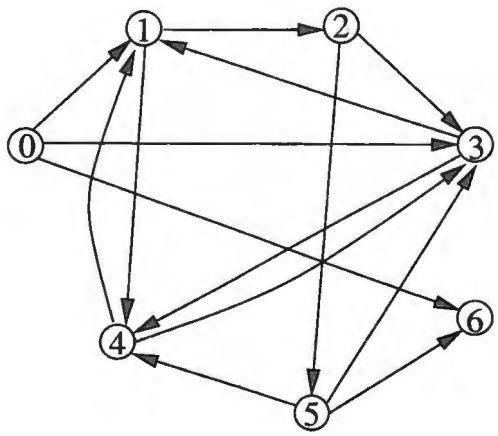


Figure 9: The Adleman't graph.

G, the vertices of G are represented by single-stranded DNA oligos twenty (randomly chosen) nucleotides long. If an edge starts at vertex v_1 and ends at vertex v_2, represent it by a 20-nucleotide sequence of single-stranded DNA as follows. The first ten nucleotides of the edge oligo are complementary to the last ten nucleotides of the vertex oligo representing v_1, and the second ten nucleotides of the edge oligo are complementary to the first ten nucleotides of the vertex oligo representing v_2 (see Fig. 10).
A path $e_1 \cdots e_k$ of length k in G is represented by a double-stranded DNA molecule of length $20k$ base pairs with ten nucleotides (single-stranded) overhang from each end. So if a Hamiltonian path exists, then it has to be

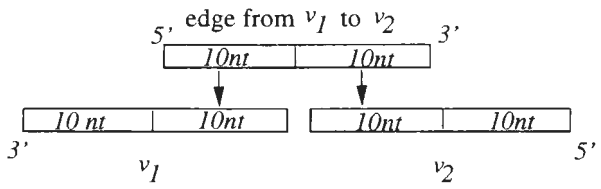


Figure 10: Joining two vertex oligos with an edge oligo

represented by a double stranded molecule of length $20n$ where n is the number of vertices. The algorithm that Adleman used for solving HPP is the following:

1. *Generate paths in the graph.* This step was done by placing the sets of olygos (large quantities of them) representing the vertices and the edges of the graph in a common mix. After the complementary parts have anealed, a ligase enzyme is applied such that the open nicks are connected (as was described in Section 2.2.3).

2. *Extract paths that begin with vertex v_{in} and end with vertex v_{out}.* This step is obtained by performing a PCR on the set of molecules obtained from the previous step with complements of the oligos representing the vertices v_{in} and v_{out} as primers. As explained in Section 2.2 with this step the molecules representing paths that begin with v_{in} and end with v_{out} are amplified in such a way that the rest of the molecules in the mix are undetectable.

3. *Extract paths that contain exactly n vertices.* From the mix generated in the previous step, molecules that are exactly $20n$ in length are extracted through gel electrophoresis.

4. *For each vertex v in G, reject all paths that do not contain v.* This was the most time consuming step. The operation was performed through a special biomolecular protocol that uses oligos complementary to the vertices and magnetic beads. The description of this protocol is beyond the scope of this article, and we refer the reader to [1, 4] for more details.

The problem that was chosen for this experiment was a well known NP-complete problem that is generally "intractable" in the sense that for a

relatively modest size of the graph, with any known algorithm, an impractical computer time is needed for solution. The Adleman's approach to this problem is not much different then a brute force search but the use of the "massive parallelism" of DNA made it very novel. A very large number of molecules, 10^{15} or more, can be used in a single reaction and the WC-complementarity assures that in the first step of the algorithm, the sequences representing paths are formed. They are formed in a very large number, such that if a solution to the problem exists, with a high probability it will be represented within one of the formed sequences. So the nondeterminism was taken care of with the massive parallelism and the WC-complementarity.

Several theoretical models based on the lab protocols used in the Adleman's experiment can be found in the literature (see for ex. [2, 26, 31]). They all use more or less the same set of operations: *merge, separate, detect, amplify etc.* and they all can feasibly be executed by a robotic system. In [26], Lipton showed that using these operations the satisfiability problem for propositional formulas can be solved and consequently a large set of problems can be solved by DNA. In [3], it is shown how these operations can be used to break the Data Encryption Standard (DES). Approximately one gram of DNA is needed and using robotic arms (assuming each operation to last one minute) breaking DES is estimated to take five days. The most significant in the analysis for breaking DES is that the success is quite likely even with, at this point unavoidable, large number of errors within the lab protocols.

The big drawback in the Adleman and Lipton's approach is the need to have a very large pool of initial molecules that have to be generated in order to assure correct solution to the problem. In Figure 9 the graph has only 7 vertices. The first step of the algorithm requires approximately 2^7 distinct molecules to be generated in order to be sure that the solution (if exists) will be present in the mix. Working with over 10^{15} DNA molecules, generating 2^7 distinct molecules does not seem difficult. But for a larger graph, say a modest size of 200 vertices, one needs "DNA more than the weight of the Earth" (see [14]). The subsequent studies have concentrated on developing algorithms such that not necessarily all of the potential solutions are constructed at once (for ex. [29, 30]).

3.2 Splicing systems

In the literature there are now too many theoretical models of different splicing (now known also as H-) systems to be mentioned here. In this

section we will concentrate on the simplest example, the "simple" splicing system. This example might not be very interesting from the computational point of view, since the language generated by a simple splicing system is strictly locally testable, but the splicing systems started with this model and its relation to the action of the endonucleases on DNA can be explained in a vivid way.

As it was explained in Section 2.2.2, restriction enzymes (endonucleases) of type II recognize specific sites (recognition sites) of a double-stranded DNA molecule and cut the molecule (by destroying the phosphodiester bonds), often in a way that leave a small single-stranded overhang.

Once cut, molecules with complementary overhangs can join together and the phosphodiester bonds can be reestablished with an aid of a ligase. (See Section 2.2.3 and Figure 7.) What words (strings of DNA molecules) can be obtained in a single tube by allowing a set of restriction enzymes and a ligase to react? This question was first considered by T. Head [15] and then followed by extensive studies of H-systems by other researchers such as R. Freund, Gh. Paun, G. Rozenberg and others. For more general H-systems, see [31, 16, 12].

Let $\mathcal{A}$ be an alphabet, $\mathcal{M} \subseteq \mathcal{A}$, and $L \subseteq \mathcal{A}^*$, and let $\sigma_{\mathcal{M}}(L)$ be defined by $z \in \sigma_{\mathcal{M}}(L)$ if and only if there exist $x, y \in L$ such that for some $c \in \mathcal{M}$, $x = x'cx'', y = y'cy''$, and $z = x'cy''$. The set of factors of a language L is denoted with $F(L)$ i.e. $F(L) = \{x : \exists y \in L, y = z_1 x z_2$ for some $z_1, z_2 \in A^*\}$. A *simple H-system* is a triple $H = (\mathcal{A}, S, \mathcal{M})$, where $\mathcal{A}$ is a finite alphabet, $S \subseteq \mathcal{A}^*$ is a finite set of initial words, called *axioms*, and $\mathcal{M} \subseteq \mathcal{A}$. The language generated by H is $L(H) = \bigcup_{i=0}^{\infty} L^{(i)}$, where the sets $L^{(i)}$ are defined recursively by $L^{(0)} = S$ and $L^{(i+1)} = L^{(i)} \cup \sigma_{\mathcal{M}}(L^{(i)}), i \geq 0$.

In order to see that simple H-systems generate strictly locally testable languages we follow [37], and recall the definition of constants for a language. A word c is a *constant* for a language L if and only if $xcy' \in L$ whenever $xcy, x'cy' \in L$. For a given simple H-system $H = (A, S, \mathcal{M})$, it is clear that if $c \in \mathcal{M}$, then c is a constant for $L(H)$. But even more is true: if $u \in F(L(H))$ is such that $F(u) \cap \mathcal{M} \neq \emptyset$, then u is a constant for $L(H)$. For let $u = u_1 c u_2$, where $c \in \mathcal{M}$, and suppose that $xuy, x'uy' \in L(H)$. Since $c \in \mathcal{M}$, the word xu_1cu_2y' is in $\sigma_{\mathcal{M}}(L(H)) = L(H)$, (consider xu_1cu_2y and $x'u_1cu_2y'$), so $xuy' \in L(H)$. It follows that if $k = \max\{|u| : u \in F(S), F(u) \cap \mathcal{M} = \emptyset\}$, then every word of length greater than k is a constant for $L(H)$. This is the well known characterization of strictly locally testable languages [27]. (See also Theorem 4.6 in [16] and Section 7.5 in [31].)

We now return to the action of endonucleases on DNA segments. Assume

that we have a finite number of distinct double-stranded DNA segments, and an infinite supply of each segment. Assume further that we have a finite number of restriction endonucleases and a ligase. Denote by E the set of restriction endonucleases. Consider the alphabet $\mathcal{A} = \{A, C, G, T\}$. Write the sequences of double-stranded DNA segments in direction $5'$ to $3'$, and consider them as elements of $\mathcal{A}^*$. For example if we have a segment $\begin{matrix} {}^{5'}AAGCCT^{3'} \\ {}_{3'}TTCGGA_{5'} \end{matrix}$, then we will write $AAGCCT$ and $AGGCTT$. For each restriction endonuclease ϵ, consider its restriction site ρ_ϵ (again in direction $5'$ to $3'$) and denote the overhang produced by the action of the enzyme by α_ϵ. For example, for *EcoR*I, the restriction site is $\rho_{Eco\text{RI}} = GAATTC$ and the overhang is $\alpha_{Eco\text{RI}} = AATT$. Denote by S' the set of all initial DNA segments, written as words in $\mathcal{A}^*$, as described above. Let $\{\alpha_1, \ldots, \alpha_k\}$ be the set of all distinct overhangs produced by the restriction enzymes in E. For $i = 1, 2, \ldots, k$, let $\bar{\alpha}_i$ be a new symbol encoding the sequence α_i, and let $\Sigma = \{\bar{\alpha}_1, \ldots, \bar{\alpha}_k\}$. Extend $\mathcal{A}$ to $\bar{\mathcal{A}} = A \cup \Sigma$ and define a simple H-system $\bar{H} = (\bar{\mathcal{A}}, S, \Sigma)$ where

$$S = \{xu\bar{\alpha}_i u'y \mid xu\alpha_i u'y \in S' \text{and there exists } \epsilon \text{ such that } \rho_\epsilon = u\alpha_i u' \text{and } \alpha_\epsilon = \alpha_i\}.$$

It is easy to see that the language $L(\bar{H})$ generated by the simple H-system $\bar{H}$ is the set of distinct molecules produced by the action of the set of endonucleases and a ligase on the set of molecules S'.

3.3 3D Graph Structures

Generally, DNA molecules have been treated as linear strings where much of the information content is encoded in the order of nucleotides that make up the DNA. Many of these algorithms require polynomial increases in the number of steps necessary to identify a solution with increasing size of the problem. But, as noted in Section 2.2.5 it is possible to form higher order, three dimensional (3D) structures with DNA molecules. Theoretically, the use of 3D DNA structures could significantly reduce the time and steps needed to identify a solution. In fact, 3D structures allow the Hamiltonian cycle problem to be solved with a constant number of steps, regardless of the number of vertices in the graph [18, 17]. In this last section we concentrate on the well known NP-complete problem "3-vertex-colorability" and show how using 3D DNA structures can solve this problem in constant number of steps. This example is also presented in [17, 19].

Let G be a graph with vertices V and edges E. The graph G is *3-vertex-colorable* if there is a surjective (onto) function $f : V \to \{a, b, c\}$ such that if two vertices $v, w \in V$ are adjacent (connected by an edge) then $f(v) \neq f(w)$. This means that out of three colors we assign one color to each vertex such that no two adjacent vertices are colored with the same color. The n-colorability is defined similarly.

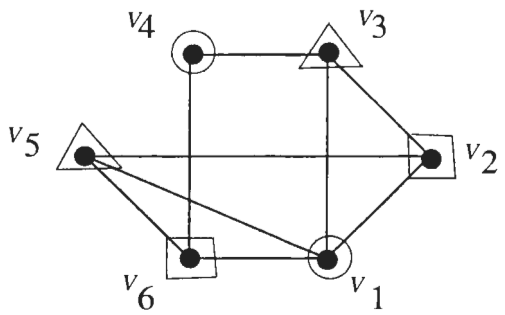


Figure 11: Coloring of a graph.

In Fig. 11, we use an example of a graph with a possible 3-vertex-coloring. In the figure, the three colors of the vertices are represented by circles, triangles and squares surrounding the vertices.

If a vertex has degree k then a k-armed branched molecule is used for its building block. For the example presented in Fig. 11, we will need one 2-armed branched molecule, four 3-armed branched molecules and one 4-armed branched molecule (see Fig. 12). The 3′ends of the k-armed branched molecules end with single stranded extensions. These extensions are 30 to 45 nucleotides long and consist of three parts, each 10 to 15 nucleotides long. The first and the third part (for example x_1 and x_3 where $x \in \{a, \ldots, i\}$ in Fig. 12) are specific encodings for the edge that is represented by the given arm of the molecule. The middle part of the encoding is the same for all arms of the vertex molecule and represents the color of the vertex. For each vertex three molecules are needed (each corresponding to one of the three possible colors of the vertex).

Each edge is a regular double-stranded molecule. The 3′ends of the molecule end with single stranded segments that are complementary to the corresponding "arm" of the vertex that is incident to the edge. Hence, the first and the third part of the encoding at the 3′end are $\bar{x}_1$ and $\bar{x}_3$ (see

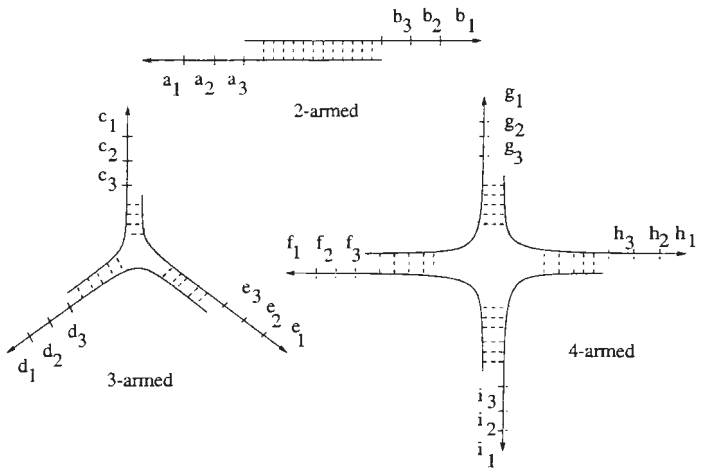


Figure 12: Building blocks for vertices.

Fig. 13) which are WC-complementary to x_1 and x_3 ($x \in \{a, \ldots, i\}$) of the corresponding "arm" of the incident vertex molecule. The middle part of the encoding $\bar{x}_2$ is complementary to the color of the incident vertex, but here, the color sequence $\bar{x}_2$ at one 3'end is different from the color sequence $\bar{y}_2$ at the other 3'end.

Figure 13: Edge building block.

For each edge we have exactly six double stranded molecules, each representing a pair of distinct colors at the endpoints of the edge molecule.

To form the graph, all edge molecules and all vertex building blocks are combined and their compatible ends are allowed to form double-stranded DNA. Once formed, the edges are locked together by sealing all open "nicks" in the DNA strands with DNA ligase. It is clear from the encoding of the problem that for a given graph G, a graph structure can be formed by vertex building blocks and edge molecules if and only if G is 3-vertex-colorable or 2-vertex-colorable.

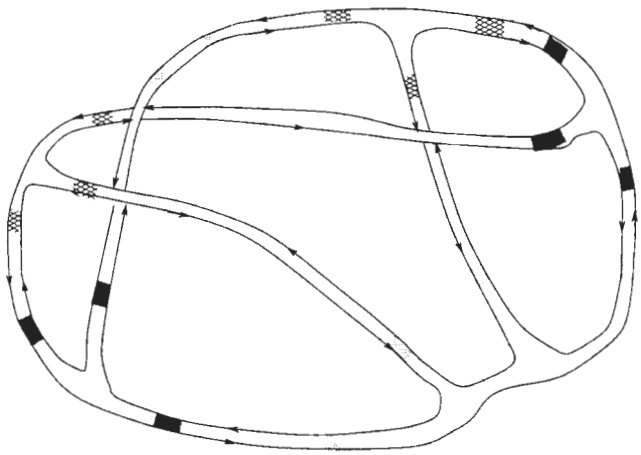


Figure 14: DNA graph structure of Fig. 11. Shading represents DNA sequences corresponding to the three colors.

The algorithm for the three vertex colorability now has the following four steps:

1. Combine all vertex building blocks with all edge molecules in a single test tube and allow the complementary ends to hybridize and be ligated.

2. Remove partially formed 3D DNA structures with open ends that have not been matched. This could be done by using an exonuclease enzyme (see [4]).

3. Remove by gel electrophoresis the graphs that are larger than the original graph formed in the above steps. This step will be explained later.

4. If there are graph structures remaining in the test tube, then we conclude that the graph is 3-colorable. Note that all 2-colorable graphs also are 3-colorable. The conlusion is confirmed with a positive control experiment.

The number of laboratory steps in this procedure does not depend on the number of vertices (or edges) in the graph. Once the building blocks are formed, the procedure needs only four steps to perform.

In Fig. 14 we show a possible formation of a graph structure. The graph structure depicted forms a knot as a single stranded DNA. Such knot structures have been used to study recombination enzymes [47, 8], and similar techniques for detecting the knot structures might be useful here. There are graphs, however, that do not form a single stranded knot, but rather form links (multicomponent circles), and partially formed graphs may contain smaller knots and links as well.

During the first step of connecting vertex building blocks with edge building blocks there is a possibility that instead of the graph that we want, the coverings of the graph be formed. Covering graphs locally are same as the original graph, but they are in general larger than the original graph. These graphs can be formed since building of the graph is done using local information contained in the vertex and edge building blocks. The molecules representing covering graphs are larger and heavier than the molecules representing the graph that is sought. These molecules can be extracted by a special gel-elctrophoresis technique. In the procedure above this is done in step 3.

4 Concluding remarks

The field of DNA computing has already branched so that other biomolecules such as RNA and liposomes [11, 5] and even cells [48] are considered as potential computational tools. Recently, the chemists at New York University constructed a molecular mechanical device attributed to the properties of B-Z transition of DNA [45]. This opens a horizon for design of computational three dimensional DNA objects that can participate in biochemical processes, and at the same time perform controlled mechanical movements.

There are certainly many more ideas and issues in computing with biomolecules than what can be mentioned here. Much work remains to be done on both experimental and theoretical level. It is impossible to say which one, or whether any of the numerous theoretical models and experimental investigations will emerge as a successful new way of computing, but the knowledge that we acquire along the way will certainly be enormous. This field has become a common platform for exchanging ideas between computer science, mathematics, molecular biology, chemistry... Its development is bound to change our ideas and understanding of both, biological processes in vivo and computing.

Acknowledgement. The work is supported by National Science Foundation grant number EIA-0074808.

References

[1] L. Adleman, *Molecular computation of solutions of combinatorial problems*, *Science* **266** (1994), 1021-1024.

[2] L. Adleman, *On constructing a molecular computer*, in R. Lipton, E. Baum, editors *DNA based computers*, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, **27** Providence RI, American Mathematical Society (1996), 1-21.

[3] L. Adleman, P.W.K. Rothemund, S. Roweis, E. Winfree, *On applying molecular computation to the Data Encryption Standard*, DIMACS series in Discrete Math. and Theoretical Comp. Sci. vol **44** (1999) 31-44.

[4] F.M. Ausubel, R. Brent, R.E. Kingston, D.D. Moore, J.G. Seidman, J.A. Smith, K. Struhl, P. Wang-Iverson and S.G. Bonitz. *Current Protocols in Molecular Biology*, Greene Publishing Associates and Wiley-Interscience, New York, NY (1993).

[5] B. Bloom, C. Bancroft, *Liposome-mediated biomolecular computation* Proceedings of the 5th International Meeting on DNA Based Computers, June 14-15, 1999, MIT Cambridge, MA, 39-46.

[6] J. Chen, N. C. Seeman, *Synthesis from DNA of a molecule with the connectivity of a cube,* Nature, **350** (1991) 631-633.

[7] J.-H. Chen, N.R. Kallenbach, N.C. Seeman, *A Specific Quadrilateral Synthesized from DNA Branched Junctions*, Journal of the American Chemical Society **111**, (1989) 6402-6407.

[8] N.R. Cozzarelli. *The structure and function of DNA supercoiling and catenanes*, The Harvey Lecture Series, **87** (1993), 35-55.

[9] M. Davis, R. Sigal, E. J. Weyuker, *Computability, Complexity, and Languages*, Academic Press, 1994 (sec. edit ion).

[10] S.M. Du, H. Wang, Y.C. Tse-Dinh, N.C. Seeman, *Topological Transformations of Synthetic DNA Knots*, Biochemistry **34**, (1995) 673-682.

[11] D. Faulhammer, A.R. Cukras, R.J. Lipton, F.L. Landweber, *Molecular Computation: RNA solutions to Chess Problems* Proc. Natl. Acad. Sci. USA, vol. **97**, (2000) 1385-1389.

[12] P. Frisco, *The hyperbibliography page on DNA computing* http://www.liacs.nl/home/pier/dna.html

[13] M. Garzon, N. Jonoska, S. Karl, *Bounded Complexity of DNA Computing*, to appear in *BioSystems*.

[14] J. Hartmanis, *On the weight of a computation*, Bul. EATCS vol. **55** (1995) 136-138.

[15] T. Head, *Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors*, *Bull. Math. Biology* **49** (1987), 737-759.

[16] T. Head, Gh. Paun, D. Pixton, *Language theory and molecular genetics*, in *Handbook of formal languages, Vol.II* (G. Rozenberg, A. Salomaa editors), Springer Verlag (1997), 295-358.

[17] N. Jonoska, *3D DNA patterns and Computing* (A.Carbone, M.Gromov, P.Pruzinkiewicz editors) World Scientific Publishing Company, Singapore, (1999) 310-324.

[18] N. Jonoska, S. Karl, M. Saito, *Three dimensional DNA structures in computing*, *BioSystems*, **52** (1999) 143-153.

[19] N. Jonoska, S. Karl, M. Saito, *Creating 3-dimensional graph structures with DNA*, in [36]

[20] L. Kari, *DNA Computing: arrival of biological mathematics*, The Mathematica l Intelligencer, **19** No.2, (1997), 9-22.

[21] J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, R.L. Riolo, eds. (1997). *Proc. 2nd Annual Genetic Programming Conference*, Morgan Kaufmann.

[22] J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, R.L. Riolo, eds. (1998). *Proc. 3rd Annual Genetic Programming Conference*, Morgan Kaufmann.

[23] L. Landweber, E. Baum, editors, Proceedings of the Second Annual Meeting on *DNA Based Computers*, DIMACS series in Discrete Math. and Theoretical Comp. Sci. vol **44** (1999).

[24] L. Landweber, L. Kari, *The evolution of cellular computing: nature's solution to a computational problem*, in [34].

[25] E. Laun, K.J. Reddy, *Wet splicing systems*, DNA Based Computers III, (H. Rubin, D.H. Wood editors), DIMACS series *AMS* **48** (1999), 73-83.

[26] R. Lipton, *DNA solution of hard computational problems*, *Science* **268** (1995), 542-545.

[27] A. de Luca, A. Restivo, *A characterization of strictly locally testable languages and its applications to subsemigroups of a free semigroup*, *Information and Control* **44** (1980), 300-319.

[28] C. Mao, W. Sun, N.C. Seeman, *Construction of Borromean Rings from DNA*, Nature **386**, (1997) 137-138.

[29] N. Morimoto, M. Arita, A. Suyama, *Solid phase DNA solution to the Hamiltonian path problem*, DNA Based Computers III, (H. Rubin, D.H. Wood editors), DIMACS series *AMS* **48** (1999), 193-206.

[30] Q. Ouyang, P.D. Kaplan, S. Liu, A. Libchaber, *DNA solution to the Maximal Clique problem*, *Science* **278** (1997), 446-449.

[31] Gh. Paun, G. Rozenberg, A. Salomaa, *DNA Computing, new computing paradigms*, Springer Verlag (1998).

[32] Gh. Paun, G. Rozenberg, A. Salomaa, *Computing by splicing*, Theoretical Computer Science **168**, No.2 (1996), 321-336.

[33] Gh. Paun, *On the power of the splicing operation*, Intern. J. Computer Math **59**, (1995), 27-35.

[34] Proceedings of the Fourth Annual Meeting on *DNA Based Computers*, DIMACS Workshop, (L. Kari editor) University of Pennsylvania, June 15-19, 1998.

[35] J. Reif, *Parallel molecular computation:models and simulation*, Seventh Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA95), Santa Barbara, June 1995.

[36] H. Rubin, D. Wood, editors, Proceedings of the Third Annual Meeting on *DNA Based Computers*, DIMACS series in Discrete Math. and Theoretical Comp. Sci. vol **48** (1999).

[37] M. P. Schützenberger, *Sur Certaines Opérations de Fermeture Dans les Language Rationnels*, *Symposia Mathematica* **15** (1975), 245-253. (Also in *RAIRO Information Theory* **1** (1974), 55-61.)

[38] N.C. Seeman, private communication.

[39] N.C. Seeman, N.R. Kallenbahc, *Nucleic Acid Junctions: A successful Experiment in Macromolecular Design*, in *Molecular Structure: Chamical Reactivity and Biological Activity* (J.J. Stezowski, J.L. Huang, M.C. Shao editors) Oxford Univ. Press, Oxford (1988), 189-194.

[40] N.C. Seeman et al. *The Perils of Polynucleotides: The Experimental gap Between the Design and Assembly of Unusual DNA Structures* in [23].

[41] N.C. Seeman et al. *Gel electrophoretic analysis of DNA branched junctions,* Electrophoresis, 10 (1989) 345–354.

[42] N.C. Seeman, *Nucleic Acid Junctions and Lattices*' Journal of Theoretical Biology **99**, 237-247 (1982).

[43] N.C. Seeman, *The Design of Single-stranded Nucleic Acid Knots* Molecular Engineering, **2** (1992), 197-307.

[44] N.C. Seeman, Y. Zhang, S.M. Du and J. Chen, *Construction of DNA polyhedra and knots through symmetry minimization,* Supermolecular Stereochemistry, J. S. Siegel, ed. (1995), 27–32.

[45] N.C. Seeman, F. Liu, C. Mao, X. Yang, L. Wenzler, E. Winfree, *DNA nanotechnology: Control of 1-D and 2-D arrays and the construction of a nanomechanical device* in [34].

[46] D.W. Sumners, *Lifting the curtain: using topology to probe the hidden action of enzymes*, Notices of the AMS, **42**, No.5 (1995), 528-537.

[47] S. Wasserman, N. Cozzarelli, *Biochemical topology: applications to DNA recombination and replication*, Science, **232** (1986), 951-960.

[48] R. Weiss, T.F. Knight, *Engineered Communications for Microbial Robotics*, Proceedings of 6th International Meeting on DNA Based Computers, (Eidtors: A. Condon, G. Rozenberg) Leiden June 13-17, 2000, The Netherlands, 5-19.

[49] E. Winfree, X. Yang, N.C. Seeman, *Universal Computation via Self-assembly of DNA: Some Theory and Experiments,* in [23].

[50] E. Winfee, F. Liu, L. Wenzler, N. C. Seeman, *Design of self-assembly of two-dimensional crystals,* Nature, vol.**494** (1998) 539-544.

[51] E. Winfree, *Whiplash PCR for $O(1)$ computing* in [34].

On the Generative Power of P Systems

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1 – 764, 70700 Bucureşti, Romania
E-mail: gpaun@imar.ro

September 16, 2000

Abstract

The aim of this paper is twofold: to introduce to the reader the main ideas of Computing with Membranes (P systems), a recent branch of (theoretical) Molecular Computing, and to survey the main results about the generative power of P systems.

We present here the model, we illustrate by an example the basic definition, then we recall several variants, both dealing with symbol-objects and with string-objects. A complete bibliography of the domain, at the level of the middle of July 2000, is also provided.

1 Introduction; The Basic Idea

The P systems (initially, in [P25][1], they were called *supper-cell* systems) were introduced as a possible answer to the question whether or not the frequent statements (see, e.g., [1], [3]) that the processes which take place in an alive cell can be considered computations, that "the alive cells are computers", are just metaphoras, or a formal computing device can be abstracted from the cell functioning. As we will show below, the answer is affirmative.

Three are the fundamental features of alive cells which will be used in our computing machineries: (1) the complex compartimentation by means of a **membrane structure**, where (2) **multisets** of chemical compounds evolve according to prescribed (3) **rules**.

[1] The references of the form [P*n*] point to papers in the P system bibliography given at the end of the paper.

A *membrane structure* is a hierarchical arrangement of membranes, all of them placed in a main membrane, called the *skin* membrane. This one delimits the system from its environment. The membranes should be understood as three-dimensional baloons, but a suggestive pictorial representation is by means of planar Euler-Venn diagrams (see Figure 1). Each membrane precisely identifies a *region*, the space between it and all the directly inner membranes, if any exists. A membrane without any membrane inside is said to be *elementary*.

In the regions of a membrane structure we place *multisets* of *objects*. A multiset is a usual set with multiplicities associated with each objects, in the form of natural numbers; the meaning is that each object can appear in a number of identical copies in a given region. For the beginning, the objects are supposed to be symbols from a given alphabet (we always work with finitely many types of objects, that is, with multisets over a finite support-set).

The objects evolve by means of given *rules*, which are associated with the regions (the intuition is that each region has specific chemical reaction conditions, hence the rules from a region cannot necessarily act also elsewhere). These rules specify both object transformation and object transfer from a region to another one. The passing of an object through a membrane is called *communication*.

Here is a typical rule

$$cabb \rightarrow caad_{out}d_{in_3},$$

with the following meaning: one copy of the *catalyst* c (note that it is reproduced after the "reaction") together with one copy of object a and two copies of object b react together and produce one copy of c, two of a, and two copies of the object d; one of these latter objects is sent out of the region where the rule is applied, while the second copy is sent to the adjacently inner membrane with the label 3, if such a membrane exists; the objects c, a, a remain in the same membrane (it is supposed that they have associated the communication command *here*, but we do not explicitly write this indication); if there is no membrane with label 3 directly inside the membrane where the rule is to be applied, then the rule cannot be applied. By a command *out*, an object can be also sent outside the skin membrane, hence it leaves the system and never comes back.

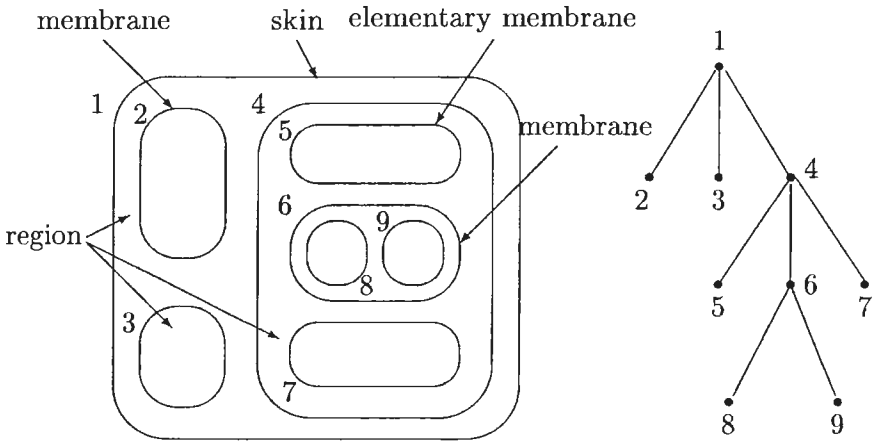


Figure 1: A membrane structure and its associated tree

Therefore, the rules perform a multiset processing; in the previous case, the multiset represented by *cabb* is subtracted from the multiset of objects in a given region, objects *caa* are added to the multiset in that region, while copies of *d* are added to the multisets in the upper and lower regions.

The rules are used *in a nondeterministic maximally parallel manner*: the objects to evolve and the rules to be applied to them are chosen in a nondeterministic manner, but no object which can evolve at a given step should remain. Sometimes, a priority relation among rules is considered, hence the rules to be used and the objects to be processed are selected in such a way that only rules which have a maximal priority among the applicable rules are used.

Other features can be considered, such as the possibility to control the membrane thickness/permebility, but we will introduce them later.

The membrane structure together with the multisets of objects and the sets of evolution rules present in its regions constitute a *P system*. The membrane structure and the objects define a *configuration* of a given P system. By using the rules as suggested above, we can define *transitions* among configurations. A sequence of transitions is called a *computation*. We accept as successful computations only the *halting* ones, those which reach a configuration where no further rule can be applied.

With a successful computation we can associate a *result*, for instance, by counting the objects present in the halting configuration in a specified

elementary membrane, i_o. More precisely, we can use a P system for solving three types of tasks: as a *generative* device (start from an initial configuration and collect all vectors of natural numbers describing the multiplicities of objects present in membrane i_o at the end of all successful computations), as a *computing* device (start with some input placed in an initial configuration and read the output at the end of a successful computation, by considering the contents of membrane i_o), and as a *decidability* device (introduce a problem in an initial configuration and look for the answer in a specified number of steps, in the contents of membrane i_o).

Of course, instead of using an elementary membrane for reading the output, we can consider the objects which leave the skin membrane and we can "compose" the result by using them: a string is obtained by arranging the symbols in the ordering of their expelling from the system, accepting each ordering in the case when several symbols leave the system at the same time. In this way, we get a language as the result of all computations. A language is also obtained when we deal with string-objects, but we will discuss this case later.

2 A Formal Definition of a P System

A membrane structure can be mathematically represented by a tree, in the natural way, or by a string of matching parentheses. The tree of the structure in Figure 1 is given in the same figure, while the parenthetic representation of that structure is the following:

$$[_1\,[_2\,]_2\,[_3\,]_3\,[_4\,[_5\,]_5\,[_6\,[_8\,]_8\,[_9\,]_9\,]_6\,[_7\,]_7\,]_4\,]_1.$$

The tree representation makes possible considering various parameters, such as the *depth* of the membrane structure, and also suggests considering membrane structures of particular types (described by linear trees, star trees, etc).

A multiset over an alphabet $V = \{a_1, \ldots, a_n\}$ is a mapping μ from V to $\mathbf{N}$, the set of natural numbers, and it can be represented by any string $w \in V^*$ such that $\Psi_V(w) = (\mu(a_1), \ldots, \mu(a_n))$, where Ψ_V is the Parikh mapping associated with V. Operations with multisets are defined in the natural manner.

With these simple prerequisites, we can define a P system (of *degree* $m, m \geq 1$) as a construct

$$\Pi = (V, T, C, \mu, w_1, \ldots, w_m, (R_1, \rho_1), \ldots, (R_m, \rho_m), i_o),$$

where:

(i) V is an alphabet; its elements are called *objects*;

(ii) $T \subseteq V$ (the *output* alphabet);

(iii) $C \subseteq V, C \cap T = \emptyset$ (*catalysts*);

(iv) μ is a membrane structure consisting of m membranes, with the membranes and the regions labeled in a one-to-one manner with elements of a given set H; in this section we use the labels $1, 2, \ldots, m$;

(v) $w_i, 1 \leq i \leq m$, are strings representing multisets over V associated with the regions $1, 2, \ldots, m$ of μ;

(vi) $R_i, 1 \leq i \leq m$, are finite sets of *evolution rules* over V associated with the regions $1, 2, \ldots, m$ of μ; ρ_i is a partial order relation over R_i, $1 \leq i \leq m$, specifying a *priority* relation among rules of R_i.
An evolution rule is a pair (u, v), which we will usually write in the form $u \to v$, where u is a string over V and $v = v'$ or $v = v'\delta$, where v' is a string over $\{a_{here}, a_{out}, a_{in_j} \mid a \in V, 1 \leq j \leq m\}$, and δ is a special symbol not in V. The length of u is called *the radius* of the rule $u \to v$.

(vii) i_o is a number between 1 and m.

When presenting the evolution rules, the indication "here" is in general omitted.

If Π contains rules of radius greater than one, then we say that Π is a system *with cooperation*. Otherwise, it is a *non-cooperative* system. A particular class of cooperative systems is that of *catalytic* systems: the only rules of a radius greater than one are of the form $ca \to cv$, where $c \in C, a \in V - C$, and v contains no catalyst; moreover, no other evolution rules contain catalysts (there is no rule of the form $c \to v$ or $a \to v_1 c v_2$, for $c \in C$).

The $(m+1)$-tuple $(\mu, w_1, \ldots, w_m)$ constitutes the *initial configuration* of Π. In general, any sequence $(\mu', w'_{i_1}, \ldots, w'_{i_k})$, with μ' a membrane structure obtained by removing from μ all membranes different from $i_1, \ldots, i_k$ (of course, the skin membrane is not removed), with w'_j strings over V, $1 \leq j \leq k$, and $\{i_1, \ldots, i_k\} \subseteq \{1, 2, \ldots, m\}$, is called a *configuration* of Π.

It should be noted the important detail that the membranes preserve the initial labeling in all subsequent configurations; in this way, the correspondence between membranes, multisets of objects, and sets of evolution rules is well specified by the subscripts of these elements.

For two configurations $C_1 = (\mu', w'_{i_1}, \ldots, w'_{i_k})$, $C_2 = (\mu'', w''_{j_1}, \ldots, w''_{j_l})$, of Π we write $C_1 \Longrightarrow C_2$, and we say that we have a *transition* from C_1 to

C_2, if we can pass from C_1 to C_2 by using the evolution rules appearing in $R_{i_1}, \ldots, R_{i_k}$ in the following manner (rather than a completely cumbersome formal definition we prefer an informal one, explained by examples).

Consider a rule $u \to v$ in a set R_{i_t}. We look to the region of μ' associated with the membrane i_t. If the objects mentioned by u, with the multiplicities at least as large as specified by u, appear in w'_{i_t}, then these objects can evolve according to the rule $u \to v$. The rule can be used only if no rule of a higher priority exists in R_{i_t} and can be applied at the same time with $u \to v$. More precisely, we start to examine the rules in the decreasing order of their priority and assign objects to them. A rule can be used only when there are copies of the objects whose evolution it describes and which were not "consumed" by rules of a higher priority and, moreover, there is no rule of a higher priority, irrespective which objects it involves, which is applicable at the same step. Therefore, all objects to which a rule *can* be applied *must* be the subject of a rule application. All objects in u are "consumed" by using the rule $u \to v$.

The result of using the rule is determined by v. If an object appears in v in the form a_{here}, then it will remain in the same region i_t. If an object appears in v in the form a_{out}, then a will exit the membrane i_t and will become an element of the region which surrounds membrane i_t. In this way, it is possible that an object leaves the system: if it goes outside the skin of the system, then it never comes back. If an object appears in the form a_{in_q}, then a will be added to the multiset from membrane q, providing that a is adjacent to the membrane q. If a_{in_q} appears in v and membrane q is not one of the membranes delimiting "from below" the region i_t, then the application of the rule is not allowed.

If the symbol δ appears in v, then membrane i_t is removed (we say *dissolved*) and at the same time the set of rules R_{i_t} (and its associated priority relation) is removed. The multiset from membrane i_t is added (in the sense of multisets union) to the multiset associated with the region which was directly external to membrane i_t. We do not allow the dissolving of the skin membrane, because this means that the whole "cell" is lost, we do no longer have a correct configuration of the system.

All these operations are performed in parallel, for all possible applicable rules $u \to v$, for all occurrences of multisets u in the regions associated with the rules, for all regions at the same time. No contradiction appears because of multiple membrane dissolving, or because simultaneous appearance of symbols of the form a_{out} and δ. If at the same step we have a_{in_i} outside a membrane i and δ inside this membrane, then, because of the simultaneity

of performing these operations, again no contradiction appears: we assume that a is introduced in membrane i at the same time when it is dissolved, thus a will remain in the region surrounding membrane i; that is, from the point of view of a, the effect of a_{in_i} in the region outside membrane i and δ in membrane i is a_{here}.

A sequence of transitions between configurations of a given P system Π is called a *computation* with respect to Π. A computation is *successful* if and only if it halts, that is, there is no rule applicable to the objects present in the last configuration, and, if the membrane i_o is present as an elementary one in the last configuration of the computation. (Note that the output membrane was not necessarily an elementary one in the initial configuration.) The result of a successful computation is $\Psi_T(w)$, where w describes the multiset of objects from T present in the output membrane in a halting configuration. The set of such vectors $\Psi_T(w)$ is denoted by $Ps(\Pi)$ (from "Parikh set") and we say that it is *generated* by Π.

3 An Example

We illustrate the notions introduced in the previous section by considering the following P system of degree 4:

$$\Pi_1 = (V, T, C, \mu, w_1, w_2, w_3, w_4, (R_1, \rho_1), (R_2, \rho_2), (R_3, \rho_3), (R_4, \rho_4), 4),$$
$$V = \{a, b, b', c, e, f\},\ T = \{e\},\ C = \emptyset,$$
$$\mu = [_1[_2[_3\,]_3[_4\,]_4]_2]_1,$$
$$w_1 = \lambda,\ R_1 = \emptyset,\ \rho_1 = \emptyset,$$
$$w_2 = \lambda,\ R_2 = \{b' \to b,\ b \to be_{in_4},\ r_1 : ff \to f,\ r_2 : f \to \delta\},\ \rho_2 = \{r_1 > r_2\},$$
$$w_3 = af,\ R_3 = \{a \to ab',\ a \to b'\delta,\ f \to ff\},\ \rho_3 = \emptyset,$$
$$w_4 = \lambda,\ R_4 = \emptyset,\ \rho_4 = \emptyset.$$

The initial configuration of the system is presented in Figure 2. No object is present in membrane 2, hence no rule can be applied here. The only possibility is to start in membrane 3, using the objects a, f, present in one copy each. By using the rules $a \to ab', f \to ff$, in parallel for all occurrences of a and f currently available, after n steps, $n \geq 0$, we get n occurrences of b' and 2^n occurrences of f. In any moment, instead of $a \to ab'$ we can use $a \to b'\delta$ (note that we always have only one copy of a). In that moment we have $n+1$ occurrences of b' and 2^{n+1} occurrences of f and we dissolve membrane 3.

The rules of the former membrane 3 are lost, the rules of membrane 2 can now be applied to the objects left free in its region. Due to the priority relation, we have to use the rule $ff \to f$ as much as possible. In one step, we pass from b'^{n+1} to b^{n+1}, while the number of f occurrences is divided by two. In the next step, $n+1$ occurrences of e are introduced in membrane 4: each occurrence of the symbol b introduces one occurrence of e. At the same time, the number of f occurrences is divided again by two.

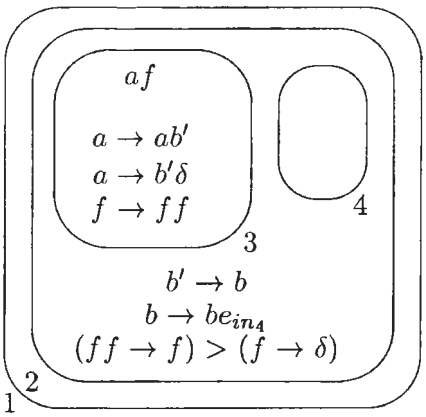


Figure 2: A P system generating $n^2, n \geq 1$

We can continue. At each step, further $n+1$ occurrences of e are introduced in the output membrane. This can be done $n+1$ steps: n times when the rule $ff \to f$ is used (thus diminishing the number of f occurrences to one), and one when using the rule $f \to \delta$ (it may now be used). In this moment, membrane 2 is dissolved, which entails the fact that its rules are removed. No further step is possible, the computation stops.

In total, we have $(n+1) \cdot (n+1)$ copies of e in membrane 4, for some $n \geq 0$, that is,

$$Ps(\Pi) = \{(n^2) \mid n \geq 1\}.$$

4 The Power of Symbol-Objects P Systems

In this section we recall some results about the generative power of some variants of P systems working with symbol-objects. In many cases, characterizations of recursively enumerable sets of vectors of natural numbers (their family is denoted by $PsRE$) are obtained.

However, before giving these results, we will specify some further ingredients which can be used in a P system. They are in general introduced in the aim of obtaining more "realistic" systems. For instance, instead the powerful command in_j, which indicates the target of the destination membrane, we can consider weaker communication commands. The weakest one is to add no label to *in*: if an object a_{in} is introduced in some region of a system, then a will go to any of the adjacent lower membranes, nondeterministically chosen; if no inner membrane exists, then a rule which introduces a_{in} cannot be used.

An intermediate possibility is to associate both with objects and membranes *electrical charges*, indicated by $+, -, 0$ (positive, negative, neutral). The charges of membranes are given in the initial configuration and are not changed during computations, the charge of objects are given by the evolution rules, in the form $a \to b^+d^-$. A charged object will immediately go into one of the directly lower membrane of the opposite polarization, nondeterministically chosen, the neutral objects remain in the same region or will exit it, according to the command *here, out* associated with it.

Moreover, besides the action δ we can also consider an opposite action, denoted by τ, in order to control the membrane thickness (hence permeability). This is done as follows. Initially, all membranes are considered of thickness 1. If a rule in a membrane of thickness 1 introduces the symbol τ, then the membrane becomes of thickness 2. A membrane of thickness 2 does not become thicker by using further rules which introduce the symbol τ, but no object can enter or exit it. If a rule which introduces the symbol δ is used in a membrane of thickness 1, then the membrane is dissolved; if the membrane had thickness 2, then it returns to thickness 1. If at the same step one uses rules which introduce both δ and τ in the same membrane, then the membrane does not change its thickness. These actions of the symbols δ, τ are illustrated in Figure 3.

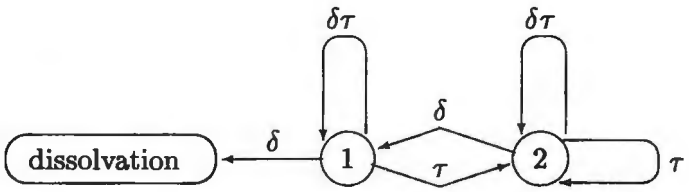


Figure 3: The effect of actions δ, τ

No object can be communicated through a membrane of thickness two, hence rules which introduce commands *out, in* requesting such communications cannot be used. However, the communication has priority over changing the thickness: if at the same step an object should be communicated and a rule introduces the action τ, then the object is communicated and "afterthat" the membrane changes the thickness.

Also a variant of catalysts can be considered, with a "short term memory". Such catalysts (we call them *bi-stable*) have two states each, c and $\overline{c}$, and they can enter rules of the forms $ca \to \overline{c}v, \overline{c}a \to cv$ (always changing from c to $\overline{c}$ and back).

Consider now some notations. The family of sets of vectors of natural numbers $Ps(\Pi)$ generated by P systems with priority, catalysts, and the actions δ, τ, and of degree at most $m, m \geq 1$, using target indications of the form *here, out,* in_j, is denoted by $NP_m(Pri, Cat, tar, \delta, \tau)$; when one of the features $\alpha \in \{Pri,\ Cat,\ \delta, \tau\}$ is not present, we replace it with $n\alpha$. When the number of membranes is not bounded, we replace the subscript m with $*$. If we use commands *here, out, in*, then we write i/o, and if use electrical charges, then we write $\pm$ instead of *tar*, respectively. We also write $2Cat$ instead of Cat when using bi-stable catalysts instead of usual catalysts.

Proofs of the following results can be found in [P25], [P27], [P31]:

Theorem 1. $NP_2(Pri, Cat, tar, n\delta, n\tau) = NP_*(nPri, Cat, \pm, \delta, \tau) = NP_1(nPri, 2Cat,\ i/o, n\delta, n\tau) = PsRE$.

The proof of the equality $NP_*(nPri, Cat, \pm, \delta, \tau) = PsRE$ from [P27] gives no bound on the number of membranes, so the problem whether or not the degree of systems of the type involved in this equality gives an infinite hierarchy is *open.*

It is also *open* the problem of the size of the family $NP_*(nPri, Cat, i/o, \delta, \tau)$. Note that in the case of using bi-stable catalysts the hierarchy on the number of membranes collapses at the first level, which shows the excessive power of this feature.

Further results about the power of symbol-object P systems can be found in the papers given in the bibliography; an early survey of such results can be found in [P26]

5 P Systems with String-Objects

As we have mentioneed in the introduction, is also possible (this was considered already in [P25]) to work with objects described by strings. The evolution rules should then be string processing rules, such as rewriting and splicing rules. As a result of a computation we can either consider the language of all strings computed by a system, or the number of strings produced by a system and "stored" in a specified membrane. In the first case one works with usual sets of strings (languages), not with multisets (each string is supposed to appear in exactly one copy), while in the latter case we have to work with multisets. We start by considering the set case.

5.1 Rewriting P Systems

We consider here the case of using string-objects processed by rewriting. Always we use only context-free rules, having associated target indications. Thus, the rules of our systems are of the form $(X \to v; tar)$, where $X \to v$ is a context-free rule over a given alphabet and $tar \in \{here, out, in, in_j\}$, with the obvious meaning: the string produced by using this rule will go to the membrane indicated by tar (j is the label of a membrane). As above, we can also use priority relations among rules as well as the actions δ, τ, and then the rules are written in the form $(X \to x\alpha; tar)$, with $\alpha \in \{\delta\tau\}$.

Formally, a *rewriting P system* is a construct

$$\Pi = (V, T, \mu, L_1, \ldots, L_m, (R_1, \rho_1), \ldots, (R_m, \rho_m), i_o),$$

where V is an alphabet, $T \subseteq V$ (the terminal alphabet), μ is a membrane structure with m membranes labeled with $1, 2, \ldots, m$, $L_1, \ldots, L_m$ are finite languages over V (initial strings placed in the regions of μ), $R_1, \ldots, R_m$ are finite sets of context-free evolution rules, $\rho_1, \ldots, \rho_m$ are partial order relations over $R_1, \ldots, R_m$, and i_o is the output membrane.

The language generated Π is denoted by $L(\Pi)$ and it is defined as follows: we start from the initial configuration of the system and proceed iteratively, by transition steps performed by using the rules in parallel, to all strings which can be rewritten, obeying the priority relations; at each step, each string which can be rewritten must be rewritten, but this is done like in a sequential manner, that is, only one rule is applied to each string; the actions δ, τ have the usual meaning; when the computation halts, we collect the terminal strings generated in the output membrane. We stress the fact

that each string is processed by one rule only, the parallelism refers here to processing simultaneously all available strings by all applicable rules.

We denote by $RP_m(Pri, tar, \delta, \tau)$ the family of languages generated by rewriting P systems of degree at most $m, m \geq 1$, using priorities, target indications of the form in_j and actions δ, τ; as usual, we use i/o, $nPri$, $n\delta, n\tau$ when appropriate.

In order to illustrate the way of working of a rewriting P system, we consider an example (which also proves that the family $RP_2(nPri, i/o, n\delta, n\tau)$ contains non-context-free languages):

$$\begin{aligned}
\Pi &= (\{A, B, a, b, c\}, \{a, b, c\}, [_1[_2\,]_2]_1, \emptyset, \{AB\}, (R_1, \emptyset), (R_2, \emptyset), 2), \\
R_1 &= \{(B \to cB; in)\}, \\
R_2 &= \{(A \to aAb; out),\ (A \to ab; here),\ (B \to c; here)\}.
\end{aligned}$$

It is easy to see that $L(\Pi) = \{a^n b^n c^n \mid n \geq 1\}$ (if a string $a^i A b^i c^i B$ is rewritten in membrane 2 to $a^i A b^i c^{i+1}$ and then to $a^{i+1} A b^{i+1} c^{i+1}$ and sent out, then it will never come back again in membrane 2, the computation stops, but the output membrane will remain empty).

The following result is proved in [P25] for the case of three membranes; the improvement to two membranes was given independently in [P15], [P43].

Theorem 2. $RP_2(Pri, i/o, n\delta, n\tau) = RE$.

The powerful feature of using a priority relation can be avoided at the price of using membranes with a variable thickness. This was proved first in [P43], [P46], without a limit on the number of membranes, then the result has been improved in [P10]:

Theorem 3. $RP_5(nPri, i/o, \delta, \tau) = RE$.

It is not known whether or not this result is optimal. The following assertions are also proved in [P10]:

Theorem 4. (i) $MAT \subseteq RP_2(nPri, i/o, \delta, \tau)$. (ii) $RP_3(nPri, i/o, \delta, \tau) - MAT \neq \emptyset$.
(iii) $RP_1(nPri, i/o, \delta, \tau) - CF \neq \emptyset$.

5.2 Splicing P Systems

The strings in a P system can also be processed by using the *splicing* operation introduced in [2] as a formal model of the DNA recombination under

the influence of restriction enzymes and ligases (see a comprehensive information about splicing in [4]).

Consider an alphabet V and two symbols $\#, \$$ not in V. A *splicing rule* over V is a string $r = u_1\#u_2\$u_3\#u_4$, where $u_1, u_2, u_3, u_4 \in V^*$ (V^* is the set of all strings over V). For such a rule r and for $x, y, w, z \in V^*$ we define

$$(x, y) \vdash_r (w, z) \quad \text{iff} \quad x = x_1u_1u_2x_2,\ y = y_1u_3u_4y_2,\ w = x_1u_1u_4y_2,\ z = y_1u_3u_2x_2,$$
$$\text{for some } x_1, x_2, y_1, y_2 \in V^*.$$

(One cuts the strings x, y in between u_1, u_2 and u_3, u_4, respectively, and one recombines the fragments obtained in this way.)

A *splicing P system* (of degree $m, m \geq 1$) is a construct

$$\Pi = (V, T, \mu, L_1, \ldots, L_m, R_1, \ldots, R_m),$$

where V is an alphabet, $T \subseteq V$ (the *output* alphabet), μ is a membrane structure consisting of m membranes (labeled with $1, 2, \ldots, m$), $L_i, 1 \leq i \leq m$, are languages over V associated with the regions $1, 2, \ldots, m$ of μ, $R_i, 1 \leq i \leq m$, are finite sets of *evolution rules* associated with the regions $1, 2, \ldots, m$ of μ, given in the following form: $(r; tar_1, tar_2)$, where $r = u_1\#u_2\$u_3\#u_4$ is a usual splicing rule over V and $tar_1, tar_2 \in \{here, out, in\} \cup \{in_j \mid 1 \leq j \leq m\}$.

Note that, as usual in splicing systems, when a string is present in a region of our system, it is assumed to appear in arbitrarily many copies.

A transition in Π is defined by applying the splicing rules from each region of μ, in parallel, to all possible strings from the corresponding regions, and following the target indications associated with the rules. More specifically, if x, y are strings in region i and $(r = u_1\#u_2\$u_3\#u_4; tar_1, tar_2) \in R_i$ such that we can have $(x, y) \vdash_r (w, z)$, then w and z will go to the regions indicated by tar_1, tar_2, respectively. Note that after splicing, the strings x, y are still available in region i, because we have supposed that they appear in arbitrarily many copies (an arbitrarily large number of them were spliced, arbitrarily many remain), but if a string w, z, resulting from a splicing, is sent out of region i, then no copy of it remains here.

The result of a computation consists of all strings over T which are sent out of the system at any time during the computation. We denote by $L(\Pi)$ the language of all strings of this type. We say that $L(\Pi)$ is *generated* by Π. Note that in this subsection we do not consider halting computations, but we leave the process to continue forever and we just observe it from outside and collect the terminal strings leaving the system.

We denote by $SP(tar, m, p)$ the family of languages $L(\Pi)$ generated by splicing P systems as above, of degree at most $m, m \geq 1$, and of depth at most $p, p \geq 1$. When we use the target indication *in* instead of in_j, then we replace *tar* with i/o.

In [P35] it was proved that $SP(i/o, 3, 3) = SP(tar, 3, 2) = SP(i/o, 5, 2) = RE$; these results were improved in [P24] as follows.

Theorem 5. $SP(i/o, 2, 2) = RE$.

Also the diameter of splicing rules (the vector of maximal lengths of the four strings appearing in a splicing rule) is considered in [P24]; from this point of view, some of the results in [P24] were further improved in [P11].

5.3 P Systems with Worm-Objects

In P systems with symbol-objects we work with multisets and the result of a computation is a natural number or a vector of natural numbers; in the case of string-object P systems we work with sets of strings and the result of a computation is a string. We can combine the two ideas: we can work with multisets of strings and consider as the result of a computation the number of strings present in the halting configuration in a given membrane. To this aim, we need operations with strings which can increase and decrease the number of occurrences of strings.

The following four operations were considered in [P4] (they are slight variants of the operations used in [5]):

1. *Replication.* If $a \in V$ and $u_1, u_2 \in V^+$, then $r : a \to u_1 || u_2$ is called a *replication rule.* For strings $w_1, w_2, w_3 \in V^+$ we write $w_1 \Longrightarrow_r (w_2, w_3)$ (and we say that w_1 is replicated with respect to rule r) if $w_1 = x_1 a x_2$, $w_2 = x_1 u_1 x_2$, $w_3 = x_1 u_2 x_2$, for some $x_1, x_2 \in V^*$.
2. *Splitting.* If $a \in V$ and $u_1, u_2 \in V^+$, then $r : a \to u_1 | u_2$ is called a *splitting rule.* For strings $w_1, w_2, w_3 \in V^+$ we write $w_1 \Longrightarrow_r (w_2, w_3)$ (and we say that w_1 is splitted with respect to rule r) if $w_1 = x_1 a x_2$, $w_2 = x_1 u_1$, $w_3 = u_2 x_2$, for some $x_1, x_2 \in V^*$.
3. *Mutation.* A *mutation rule* is a context-free rewriting rule, $r : a \to u$, over V. For strings $w_1, w_2 \in V^+$ we write $w_1 \Longrightarrow_r w_2$ if $w_1 = x_1 a x_2$, $w_2 = x_1 u x_2$, for some $x_1, x_2 \in V^*$.
4. *Recombination.* Consider a string $z \in V^+$ (as a *crossing-over block*) and four strings $w_1, w_2, w_3, w_4 \in V^+$. We write $(w_1, w_2) \Longrightarrow_z (w_3, w_4)$ if $w_1 = x_1 z x_2$, $w_2 = y_1 z y_2$, and $w_3 = x_1 z y_2$, $w_4 = y_1 z x_2$, for some $x_1, x_2, y_1, y_2 \in V^*$.

Note that replication and splitting increase the number of strings, mutation and recombination not; by sending strings out of the system, their number can also be decreased.

A *P system* (of degree $m, m \geq 1$) *with worm-objects* is a construct

$$\Pi = (V, \mu, A_1, \ldots, A_m, (R_1, S_1, M_1, C_1), \ldots, (R_m, S_m, M_m, C_m), i_o),$$

where:

- V is an alphabet;
- μ is a membrane structure of degree m (with the membranes labeled with $1, 2, \ldots, m$);
- $A_1, \ldots, A_m$ are multisets of a finite support over V^*, associated with the regions of μ;
- for each $1 \leq i \leq m$, R_i, S_i, M_i, C_i are finite sets of replication rules, splitting rules, mutation rules, and crossing-over blocks, respectively, given in the following forms:
  a. replication rules: $(a \rightarrow u_1 || u_2; tar_1, tar_2)$, for $tar_1, tar_2 \in \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$;
  b. splitting rules: $(a \rightarrow u_1 | u_2; tar_1, tar_2)$, for $tar_1, tar_2 \in \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$;
  c. mutation rules: $(a \rightarrow u; tar)$, for $tar \in \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$;
  d. crossing-over blocks: $(z; tar_1, tar_2)$, for $tar_1, tar_2 \in \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$;
- $i_o \in \{1, 2, \ldots, m\}$ specifies the *output membrane* of the system; it should be an elementary membrane of μ.

The $(m+1)$-tuple $(\mu, A_1, \ldots, A_m)$ constitutes the *initial configuration* of the system. By applicating the operations defined by the components $(R_i, S_i, M_i, C_i), 1 \leq i \leq m$, we can define transitions from a configuration to another one. This is done as usual in P system area, according to the following specific rules: A string which enters an operation is "consumed" by that operation, its multiplicity is decreased by one. The multiplicity of strings produced by an operation is accordingly increased. A string is processed by only one operation. For instance, we cannot apply two mutation rules, or a mutation rule and a replication one, to the same string. The strings resulting from an operation are communicated to the region specified by the target indications associated with the used rule.

The result of a halting computation consists of the number of strings in region i_o at the end of the computation. A non-halting computation provides no output. For a system Π, we denote by $N(\Pi)$ the set of numbers computed in this way. By $NWP_m, m \geq 1$, we denote the sets of numbers computed by all P systems with at most m membranes.

In [P4] it is proved that each recursively enumerable set of natural numbers (their family is denoted by nRE) can be computed by a P system as above; the result is improved in [P20], where it is proved that the hierarchy on the number of membranes collapses:

Theorem 6. $nRE = NWP_6$.

It is an *open problem* whether or not the bound 6 in this theorem can be improved; we expect a positive answer.

The use of the powerful feature of priority relations (Theorem 2) can be avoided also if we combine the rewriting with other features, of the types used in the present subsection. More precisely, let us consider P systems working with multisets of worm-objects, processed by rewriting and crossovering rules, but let us consider as the result of a computation the language of all strings present at the end of halting computations in a specified output membrane. The work of such a system is exactly as the work of a P system with worm-objects, only the way of defining the result of a computation is different.

Let us denote by $RXP_m(nPri, i/o, n\delta, n\tau), m \geq 1$, the family of languages generated by such systems with at most m membranes, using as communication commands the indications *here, out, in* (but not priorities and actions δ, τ). Somewhat expected, we get one further characterization of recursively enumerable languages (the proof can be found in [P21]).

Theorem 7. $RE = RXP_5(nPri, i/o, n\delta, n\tau)$.

It is an *open problem* whether or not the bound 5 is optimal.

6 Final Remarks

We have considered here only some of the variants of P systems with symbol-objects and with string-objects and we have recalled mainly characterizations of the family of recursively enumerable sets of vectors of natural numbers or of languages. Many other variants, several of them leading to similar results, can be found in the literature. We only mention here the generalized P systems considered in [P7], [P8], [P9], the plasmid-based systems (where

no object evolves, but only passes through membranes) [P22], the P systems with valuations [P19] (their power is not know yet), the systems also able to create membranes [P12], which lead to a characterization of Parikh sets of ET0L languages, and the systems also taking into account the energy created/consumed by each evolution rule [P32]. Many papers deal with the possibility of solving NP-complete problems in a polynomial – often linear – time, [P4], [P13], [P14], [P19], [P28], [P33], [P44], or with "implementations" (we call them "simulations") of P systems on the usual computer, [P2], [P18], [P40], [P41]. The reader is refereed to the mentioned papers for details.

Bibliography of P systems (July 2000)

[P1] A. Atanasiu, Arithmetic with membranes, *Pre-proc. of Workshop on Multiset Processing*, Curtea de Argeş, Romania, 2000.

[P2] A. V. Baranda, J. Castellanos, F. Arroyo, R. Gonzalo, Data structures for implementing P systems in silico, *Pre-proc. of Workshop on Multiset Processing*, Curtea de Argeş, Romania, 2000.

[P3] C. Calude, Gh. Păun, *Computing with Cells and Atoms*, Taylor and Francis, London, 2000 (Chapter 3: "Computing with Membranes").

[P4] J. Castellanos, Gh. Păun, A. Rodriguez-Paton, Computing with membranes: P systems with worm-objects, *IEEE Conf. SPIRE 2000*, La Coruña, Spain, and *Auckland University, CDMTCS Report* No 123, 2000 (www.cs.auckland.ac.nz/CDMTCS).

[P5] J. Dassow, Gh. Păun, On the power of membrane computing, *J. of Universal Computer Sci.*, 5, 2 (1999), 33–49 (www.iicm.edu/jucs).

[P6] J. Dassow, Gh. Păun, Concentration controlled P systems, submitted, 2000.

[P7] R. Freund, Generalized P systems, *Fundamentals of Computation Theory, FCT'99*, Iaşi, 1999 (G. Ciobanu, Gh. Păun, eds.), *LNCS* 1684, Springer, 1999, 281–292.

[P8] R. Freund, Generalized P systems with splicing and cutting/recombination, *Grammars*, 2, 3 (1999), 189–199.

[P9] R. Freund, F. Freund, Molecular computing with generalized homogeneous P systems, *Proc. Conf. DNA6* (A. Condon, G. Rozenberg, eds.), Leiden, 2000, 113–125.

[P10] R. Freund, Gh. Păun, On a hierarchy of rewriting P systems, submitted, 2000.

[P11] P. Frisco, Membrane computing based on splicing: improvements, *Pre-proc. of Workshop on Multiset Processing*, Curtea de Argeş, Romania, 2000.

[P12] M. Ito, C. Martin-Vide, Gh. Păun, A characterization of Parikh sets of ET0L languages in terms of P systems, submitted, 2000.

[P13] S. N. Krishna, Computing with simple P systems, *Pre-proc. Workshop on Multiset Processing*, Curtea de Argeş, Romania, 2000.

[P14] S. N. Krishna, R. Rama, A variant of P systems with active membranes: Solving NP-complete problems, *Romanian J. of Information Science and Technology*, 2, 4 (1999), 357–367.

[P15] S. N. Krishna, R. Rama, On the power of P systems with sequential and parallel rewriting, *Intern. J. Computer Math.*, 77, 1-2 (2000), 1–14.

[P16] S. N. Krishna, R. Rama, Computing with P systems, submitted, 2000.

[P17] S. N. Krishna, R. Rama, On simple P systems with external output, submitted, 2000.

[P18] M. Maliţa, Membrane computing in Prolog, *Pre-proc. of Workshop on Multiset Processing*, Curtea de Argeş, Romania, 2000.

[P19] C. Martin-Vide, V. Mitrana, P systems with valuations, *Pre-proc. of Workshop on Multiset Processing*, Curtea de Argeş, Romania, 2000.

[P20] C. Martin-Vide, Gh. Păun, Computing with membranes. One more collapsing hierarchy, *Bulletin of the EATCS*, to appear.

[P21] C. Martin-Vide, Gh. Păun, String objects in P systems, *Proc. of Workshop on Algebraic Systems, Formal Languages and Computations*, Kyoto, 2000, RIMS Kokyuroku Series, Kyoto Univ., 2000.

[P22] C. Martin-Vide, Gh. Păun, G. Rozenberg, Plasmid-based P systems, submitted, 2000.

[P23] A. Păun, On P systems with membrane division, submitted, 2000.

[P24] A. Păun, M. Păun, On the membrane computing based on splicing, submitted, 2000.

[P25] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61 (2000), in press, and *Turku Center for Computer Science-TUCS Report* No 208, 1998 (www.tucs.fi).

[P26] Gh. Păun, Computing with membranes. An introduction, *Bulletin of the EATCS*, 67 (Febr. 1999), 139–152.

[P27] Gh. Păun, Computing with membranes – A variant: P systems with polarized membranes, *Intern. J. of Foundations of Computer Science*, 11, 1 (2000), 167–182.

[P28] Gh. Păun, P systems with active membranes: Attacking NP complete problems, *J. Automata, Languages, and Combinatorics*, 5 (2000), in press, and *Auckland University, CDMTCS Report* No 102, 1999 (www.cs.auckland.ac.nz/CDMTCS).

[P29] Gh. Păun, Computing with membranes (P systems): Twenty six research topics, *Auckland University, CDMTCS Report* No 119, 2000 (www.cs.auckland.ac.nz/ CDMTCS).

[P30] Gh. Păun, G. Rozenberg, A. Salomaa, Membrane computing with external output, *Fundamenta Informaticae*, 41, 3 (2000), 259–266.

[P31] Gh. Păun, Y. Sakakibara, T. Yokomori, P systems on graphs of restricted forms, submitted, 1999.

[P32] Gh. Păun, Y. Suzuki, H. Tanaka, P Systems with energy accounting, submitted, 2000.

[P33] Gh. Păun, Y. Suzuki, H. Tanaka, T. Yokomori, On the power of membrane division in P systems, *Proc. of Conf. on Words, Semigroups, Combinatorics*, Kyoto, 2000.

[P34] Gh. Păun, G. Thierrin, Multiset processing by means of systems of finite state transducers, *Pre-Proc. of Workshop on Implementing Automata* WIA99, Potsdam, August 1999, Preprint 5/1999 of Univ. Potsdam (O. Boldt, H. Jürgensen, L. Robbins, eds.) XV 1-17.

[P35] Gh. Păun, T. Yokomori, Membrane computing based on splicing, *Preliminary Proc. of Fifth Intern. Meeting on DNA Based Computers* (E. Winfree, D. Gifford, eds.), MIT, June 1999, 213–227.

[P36] Gh. Păun, T. Yokomori, Simulating H systems by P systems, *Journal of Universal Computer Science*, 6, 2 (2000), 178–193 (www.iicm.edu/jucs).

[P37] Gh. Păun, S. Yu, On synchronization in P systems, *Fundamenta Informaticae*, 38, 4 (1999), 397–410.

[P38] I. Petre, A normal form for P systems, *Bulletin of the EATCS*, 67 (Febr. 1999), 165–172.

[P39] I. Petre, L. Petre, Mobile ambients and P systems, *J. Universal Computer Sci.*, 5, 9 (1999), 588–598.

[P40] Y. Suzuki, H. Tanaka, On a LISP implementation of a class of P systems, *Romanian J. of Information Science and Technology*, 3, 2 (2000).

[P41] Y. Suzuki, H. Tanaka, Artificial life and P systems, *Pre-proc. Workshop on Multiset Processing*, Curtea de Argeş, Romania, 2000.

[P42] Cl. Zandron, Cl. Ferretti, G. Mauri, Two normal forms for rewriting P systems, submitted, 2000.

[P43] Cl. Zandron, Cl. Ferretti, G. Mauri, Priorities and variable thickness of membranes in rewriting P systems, submitted, 2000.

[P44] Cl. Zandron, Cl. Ferretti, G. Mauri, Solving NP-complete problems using P systems with active membranes, submitted, 2000.

[P45] Cl. Zandron, Cl. Ferretti, G. Mauri, Using membrane features in P systems, *Pre-proc. of Workshop on Multiset Processing*, Curtea de Argeş, Romania, 2000.

[P46] Cl. Zandron, G. Mauri, Cl. Ferretti, Universality amd normal forms on membrane systems, *Proc. Intern. Workshop Grammar Systems 2000* (R. Freund, A. Kelemenova, eds.), Bad Ischl, Austria, July 2000, 61–74.

References

[1] D. Bray, Protein molecules as computational elements in living cells, *Nature*, 376 (1995), 307–312

[2] T. Head, Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors, *Bulletin of Mathematical Biology*, 49 (1987), 737–759.

[3] W. R. Loewenstein, *The Touchstone of Life. Molecular Information, Cell Communication, and the Foundations of Life*, Oxford Univ. Press, New York, Oxford, 1999.

[4] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Berlin, 1998.

[5] M. Sipper, Studying Artificial Life using a simple, general cellular model, *Artificial Life Journal*, 2, 1 (1995), 1–35.

Kapitel 2

Quantum Computing

Quantum Finite Automata *

Rūsiņš Freivalds†
Institute of Mathematics and Computer Science, University of Latvia,
Raiņa bulvāris 29, Riga, LV-1459, Latvia

Abstract

Quantum finite automata are expected to appear much sooner that full-size quantum computers. Quantum finite automata do not generalize deterministic finite automata. Their capabilities are incomparable. We construct a hierarchy of regular languages such that the current language in the hierarchy can be accepted by 1-way quantum finite automata with a probability smaller than the corresponding probability for the preceding language in the hierarchy. These probabilities converge to $\frac{1}{2}$.

1 Introduction

At his Almaden laboratory, Isaac L. Chuang and colleagues [VSBYCC00] operate one of the world's most capable quantum computers – a three-qubit machine that can search an eight-place database in two steps, twice as fast as possible with conventional computers.

However the problem to build a practical quantum computer with a reasonably large memory is still considered as a big challenge for the new century.

Hence, most probably, more attention should be given to the study of simpler models like 1-way QFAs. A 1-way quantum automaton is a very reasonable model of computation and it is easy to see how it can be implemented. The finite dimensional state-space of a QFA corresponds to a

*the paper is based on joint research with Andris Ambainis, Aija Bērziņa, Richard Bonner, Renārs Gailis, Marats Golovkins, Arnolds Ķikusts, Maksim Kravtsev, Zigmārs Rasščevskis, Madars Rikards, and Māris Valdats.

†Research supported by Contract IST-1999-11234 (QAIP) from the European Commission, and Grant No.96.0282 from the Latvian Council of Science.

system with finitely many particles. Each letter has a corresponding unitary transformation on the state-space. A classical device can read symbols from the input and apply the corresponding transformations to the quantum mechanical part.

This paper is an attempt to summarize the first results in the research directed to finding out what quantum finite automata can do and what they cannot. The research is still from over. We still have no good description of the class of the languages recognizable by 1-way quantum finite automata.

Since there have been rather many papers on quantum automata, the paper concentrates on research done on language recognition with the probability of the correct result bounded away from $\frac{1}{2}$.

2 Preliminaries

2.1 Basics of quantum computation

To explain the difference between classical and quantum mechanical world, we first consider one-bit systems. A classical bit is in one of two classical states *true* and $false$. A *probabilistic* counterpart of the classical bit can be *true* with a probability α and $false$ with probability β, where $\alpha + \beta = 1$. A *quantum bit (qubit)* is very much like to it with the following distinction. For a *qubit* α and β can be arbitrary complex numbers with the property $\|\alpha\|^2 + \|\beta\|^2 = 1$. If we observe a qubit, we get *true* with probability $\|\alpha\|^2$ and $false$ with probability $\|\beta\|^2$, just like in probabilistic case. However, if we modify a quantum system without observing it (we will explain what this means), the set of transformations that one can perform is larger than in the probabilistic case. This is where the power of quantum computation comes from.

More generally, we consider quantum systems with m basis states. We denote the basis states $|q_1\rangle$, $|q_2\rangle$, ..., $|q_m\rangle$. Let ψ be a linear combination of them with complex coefficients

$$\psi = \alpha_1 |q_1\rangle + \alpha_2 |q_2\rangle + \ldots + \alpha_m |q_m\rangle .$$

Then, l_2 norm of ψ is

$$\|\psi\| = \sqrt{|\alpha_1|^2 + |\alpha_2|^2 + \ldots + |\alpha_m|^2}.$$

The state of a quantum system can be any ψ with $\|\psi\| = 1$. ψ is called *superposition* of $|q_1\rangle$, ..., $|q_m\rangle$. α_1, ..., α_m are called *amplitudes* of $|q_1\rangle$,

..., $|q_m\rangle$. We use $l_2(Q)$ to denote the vector space consisting of all linear combinations of $|q_1\rangle$, ..., $|q_m\rangle$.

Allowing arbitrary complex amplitudes is essential for physics. However, it is not important for quantum computation. Anything that can be computed with complex amplitudes can be done with only real amplitudes as well[BV97]. However, it is important that *negative* amplitudes are allowed. Starting from this place, all amplitudes are assumed to be (possibly negative) reals.

There are two types of transformations that can be performed on a quantum system. The first type are unitary transformations. A unitary transformation is a linear transformation U on $l_2(Q)$ that preserves l_2 norm. (This means that any ψ with $\|\psi\| = 1$ is mapped to ψ' with $\|\psi'\| = 1$.)

Second, there are measurements. The simplest measurement is observing $\psi = \alpha_1 |q_1\rangle + \alpha_2 |q_2\rangle + \ldots + \alpha_m |q_m\rangle$ in the basis $|q_1\rangle, \ldots, |q_m\rangle$. It gives $|q_i\rangle$ with probability α_i^2. ($\|\psi\| = 1$ guarantees that probabilities of different outcomes sum to 1.) After the measurement, the state of the system changes to $|q_i\rangle$ and repeating the measurement gives the same state $|q_i\rangle$.

In this paper, we also use *partial measurements*. Let $Q_1, \ldots, Q_k$ be pairwise disjoint subsets of Q such that $Q_1 \cup Q_2 \cup \ldots \cup Q_k = Q$. Let E_j, for $j \in \{1, \ldots, k\}$, denote the subspace of $l_2(Q)$ spanned by $|q_j\rangle$, $j \in Q_i$. Then, a *partial measurement* w.r.t. $E_1, \ldots, E_k$ gives the answer $\psi \in E_j$ with probability $\sum_{i \in Q_j} \alpha_i^2$. After that, the state of the system collapses to the projection of ψ to E_j. This projection is $\psi_j = \sum_{i \in Q_j} \alpha_i |q_i\rangle$.

2.2 Quantum finite automata

Quantum finite automata were introduced twice. First this was done by C. Moore and J.P.Crutchfield [MC97]. Later in a different and non-equivalent way these automata were introduced by A. Kondacs and J. Watrous [KW97].

The first definition just mimics the definition of 1-way probabilistic finite automata only substituting *stochastic* matrices by *unitary* ones. We use a more elaborated definition [KW97].

A QFA is a tuple $M = (Q; \Sigma; V; q_0; Q_{acc}; Q_{rej})$ where Q is a finite set of states, Σ is an input alphabet, V is a transition function, $q_0 \in Q$ is a starting state, and $Q_{acc} \subset Q$ and $Q_{rej} \subset Q$ are sets of accepting and rejecting states. The states in Q_{acc} and Q_{rej} are called *halting states* and the states in $Q_{non} = Q - (Q_{acc} \cup Q_{rej})$ are called *non halting states*. κ and \$ are symbols that do not belong to Σ. We use κ and \$ as the left and the right endmarker, respectively. The *working alphabet* of M is $\Gamma = \Sigma \cup \{\kappa; \$\}$.

The transition function V is a mapping from $\Gamma \times l_2(Q)$ to $l_2(Q)$ such that,

for every $a \in \Gamma$, the function $V_a : l_2(Q) \to l_2(Q)$ defined by $V_a(x) = V(a, x)$ is a unitary transformation.

The computation of a QFA starts in the superposition $|q_0\rangle$. Then transformations corresponding to the left endmarker κ, the letters of the input word x and the right endmarker \$ are applied. The transformation corresponding to $a \in \Gamma$ consists of two steps.

1. First, V_a is applied. The new superposition ψ' is $V_a(\psi)$ where ψ is the superposition before this step.

2. Then, ψ' is observed with respect to $E_{acc}, E_{rej}, E_{non}$ where $E_{acc} = span\{|q\rangle : q \in Q_{acc}\}$, $E_{rej} = span\{|q\rangle : q \in Q_{rej}\}$, $E_{non} = span\{|q\rangle : q \in Q_{non}\}$ (see section 2.1).

If we get $\psi' \in E_{acc}$, the input is accepted. If we get $\psi' \in E_{rej}$, the input is rejected. If we get $\psi' \in E_{non}$, the next transformation is applied.

We regard these two transformations as reading a letter a. We use V'_a to denote the transformation consisting of V_a followed by projection to E_{non}. This is the transformation mapping ψ to the non-halting part of $V_a(\psi)$. Also, we use ψ_y to denote the non-halting part of QFA's state after reading the left endmarker κ and the word $y \in \Sigma^*$.

The early work on 1-way quantum finite automata (QFAs) has mainly considered 3 questions:

1. What is the class of languages recognized by QFAs?
2. How does the size of QFAs (the number of states) compare to the size of deterministic (probabilistic) automata?
3. What accepting probabilities can be achieved?

3 Quantum vs. deterministic and probabilistic

The first results in this direction were obtained by Kondacs and Watrous [KW97].

Theorem 3.1 [KW97]

1. *All languages recognized by 1-way QFAs are regular.*
2. *There is a regular language that cannot be recognized by a 1-way QFA with probability* $\frac{1}{2} + \epsilon$ *for any* $\epsilon > 0$.

Brodsky and Pippenger [BP99] generalized the second part of Theorem 3.1 by showing that any language satisfying a certain property is not recognizable by a QFA.

Theorem 3.2 [BP99] *Let L be a language and M be its minimal automaton (the smallest DFA recognizing L). Assume that there is a word x such that M contains states q_1, q_2 satisfying:*

1. *$q_1 \neq q_2$,*
2. *If M starts in the state q_1 and reads x, it passes to q_2,*
3. *If M starts in the state q_2 and reads x, it passes to q_2, and*
4. *There is a word y such that if M starts in q_2 and reads y, it passes to q_1,*

then L cannot be recognized by any 1-way quantum finite automaton (Fig.1).

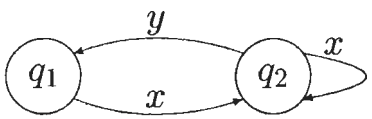


Figure 1: Conditions of theorem 3.2

A language L with the minimal automaton not containing a fragment of Theorem 3.2 is called *satisfying the partial order condition* [MT69]. [BP99] conjectured that any language satisfying the partial order condition is recognizable by a 1-way QFA. We will return to this conjecture below.

4 Complexity

Another most important issue was the complexity of automata. For 1-way finite automata, the most natural complexity measure is the number of states in the automaton. We can follow the proof in [KW97] that any language recognized by a 1-way QFA is regular step by step and add complexity bounds to it. Then, we get

Theorem 4.1 [KW97] *Let L be a language recognized by a 1-way quantum finite automaton with n states. Then it can be recognized by a 1-way deterministic automaton with $2^{O(n)}$ states.*

So, transforming a quantum automaton into a classical automaton can cause an exponential increase in its size. The results by A. Ambainis and R. Freivalds [AF98] show that, indeed, 1-way QFAs can be exponentially smaller than their classical counterparts.

Let p be a prime. We consider the language $L_p = \{a^i | i \text{ is divisible by } p\}$. It is easy to see that any deterministic 1-way finite automaton recognizing L_p has at least p states. However, there is a much more efficient QFA!

Theorem 4.2 [AF98] *For any $\epsilon > 0$, there is a QFA with $O(\log p)$ states recognizing L_p with probability $1 - \epsilon$.*

Proof. First, we construct an automaton accepting all words in L with probability 1 and accepting words not in L with probability at most 7/8. Later, we will show how to increase the probability of correct answer to $1-\epsilon$ for an arbitrary constant $\epsilon > 0$.

Let U_k, for $k \in \{1, \ldots, p-1\}$ be a quantum automaton with a set of states $|Q| = \{q_0, q_1, q_{acc}, q_{rej}\}$, a starting state $|q_0\rangle$, $Q_{acc} = \{q_{acc}\}$, $Q_{rej} = \{q_{rej}\}$. The transition function is defined as follows. Reading a maps $|q_0\rangle$ to $\cos\phi|q_0\rangle + i\sin\phi|q_1\rangle$ and $|q_1\rangle$ to $i\sin\phi|q_0\rangle + \cos\phi|q_1\rangle$ where $\phi = \frac{2\pi k}{p}$. (It is easy to check that this transformation is unitary.) Reading the right endmarker \$ maps $|q_0\rangle$ to $|q_{acc}\rangle$ and $|q_1\rangle$ to $|q_{rej}\rangle$.

Lemma 4.1 [AF98] *After reading a^j, the state of U_k is*

$$\cos\left(\frac{2\pi jk}{p}\right)|q_0\rangle + i\sin\left(\frac{2\pi jk}{p}\right)|q_1\rangle.$$

Proof. By induction. □

If j is divisible by p, then $\frac{2\pi jk}{p}$ vis a multiple of 2π, $\cos(\frac{2\pi jk}{p}) = 1$, $\sin(\frac{2\pi jk}{p}) = 0$, reading a^j maps the starting state $|q_0\rangle$ to $|q_0\rangle$ and the right endmarker \$ maps it to $|q_{acc}\rangle$. Therefore, all automata U_k accept words in L with probability 1.

For a word $a^j \notin L$, call U_k "good" if U_k rejects a^j with probability at least 1/2.

Lemma 4.2 [AF98] *For any $a^j \notin L$, at least $(p-1)/2$ of all U_k are "good".*

Proof. The superposition of U_k after reading a^j is $\cos(\frac{2\pi jk}{p})|q_0\rangle +$
$+\sin(\frac{2\pi jk}{p})|q_1\rangle$. This is mapped to $\cos(\frac{2\pi jk}{p})|q_{acc}\rangle + \sin(\frac{2\pi jk}{p})|q_{rej}\rangle$ by the right endmarker. Therefore, the probability of U_k accepting a^i is $\cos^2(\frac{2\pi jk}{p})$.

$\cos^2(\frac{2\pi jk}{p}) \leq 1/2$ if and only if $|\cos(\frac{2\pi jk}{p})| \leq 1/\sqrt{2}$. This happens if and only if $\frac{2\pi jk}{p}$ is in $[2\pi l + \pi/4, 2\pi l + 3\pi/4]$ or in $[2\pi l + 5\pi/4, 2\pi l + 7\pi/4]$ for some $l \in \mathbf{N}$.

$\frac{2\pi(jk \bmod p)}{p} \in [\pi/4, 3\pi/4]$ if and only if $\frac{2\pi jk}{p} \in [2\pi l + \pi/4, 2\pi l + 3\pi/4]$ for some l. p is a prime and j is relatively prime with p. Therefore, $j \bmod p$, $2j \bmod p, \ldots, (p-1)j \bmod p$ are just $1, 2, \ldots, p-1$ in different order. Hence, it is enough to count k such that $\frac{2\pi k}{p} \in [\pi/4, 3\pi/4]$ or $\frac{2\pi k}{p} \in [5\pi/4, 7\pi/4]$.

We do the counting for $p = 8m + 1$. (Other cases are similar.) Then $\frac{2\pi k}{p} \in [\pi/4, 3\pi/4]$ if and only if $m + 1 \leq k \leq 3m$ and $\frac{2\pi k}{p} \in [5\pi/4, 7\pi/4]$ if and only if $5m+1 \leq k \leq 7m$. Together, this gives us $4m = (p-1)/2$ "good" k's. □

Next, we consider sequences of $\lceil 8 \ln p \rceil$ k's. A sequence is *good* for a^j if at least $1/4$ of all its elements are good for a^j.

Lemma 4.3 [AF98] *There is a sequence of length* $\lceil 8 \ln p \rceil$ *which is good for all* $a^j \notin L$.

Proof. First, we show that at most $1/p$ fraction of all sequences is not good for any fixed $a^j \notin L$.

We select a sequence randomly by selecting each of its elements uniformly at random from $\{1, \ldots, p-1\}$. The probability of selecting a good k in each step is at least $1/2$. By Chernoff bounds, the probability that less than $1/4 = 1/2 - 1/4$ fraction of all elements is good is at most

$$e^{-2(1/4)^2 8 \ln p} = \frac{1}{p}.$$

Hence, the fraction of sequences which are bad for at least one $j \in \{1, 2, \ldots, p-1\}$ is at most $(p-1)/p$ and there is a sequence which is good for all $j \in \{1, \ldots, p-1\}$. This sequence is good for $a^j \notin L$ with $j > p$ as well because any U_k returns to the starting state after reading a^p and, hence, works in the same way on a^j and $a^{j \bmod p}$. □

First, we show that at most $1/p$ fraction of all sequences is not good for any fixed $a^j \notin L$.

We select a sequence randomly by selecting each of its elements uniformly at random from $\{1, \ldots, p-1\}$. The probability of selecting a good k in each step is at least $1/2$. By Chernoff bounds, the probability that less than $1/4 = 1/2 - 1/4$ fraction of all elements is good is at most

$$e^{-2(1/4)^2 8 \ln p} = \frac{1}{p}.$$

Hence, the fraction of sequences which are bad for at least one $j \in \{1, 2, \ldots, p-1\}$ is at most $(p-1)/p$ and there is a sequence which is good for all $j \in \{1, \ldots, p-1\}$. This sequence is good for $a^j \notin L$ with $j > p$ as well because any U_k returns to the starting state after reading a^p and, hence, works in the same way on a^j and $a^{j \bmod p}$. □

Next, we use a good sequence $k_1, \ldots, k_{\lceil 8 \ln p \rceil}$ to construct a quantum automaton recognizing L_p. The automaton consists of $U_{k_1}, U_{k_2}, \ldots, U_{k_{\lceil 8 \ln p \rceil}}$ and a distinguished starting state. Upon reading the left endmarker κ, it passes from the starting state to a superposition where $|q_0\rangle$ states of all U_{k_l} have equal amplitudes.

Words in L are always accepted because all U_k accept them. For any $a^j \notin L$, at least $1/4$ of the sequence is good. This means that at least $1/4$ of all U_{k_l} reject it with probability at least $1/2$ and the total probability of rejecting any $a^j \notin L$ is at least $1/8$.

It is described in more detail in [AF98] how to increase the probability of correct answer to $1 - \epsilon$ for an arbitrary $\epsilon > 0$.

Next, we compare quantum and probabilistic finite automata. Generally, 1-way probabilistic finite automata can recognize some languages with the number of states being close to the logarithm of the number of states needed by a deterministic automaton [Fre82],[Amb96]. However, this is not the case with L_p. Here, adding probabilism does not help to decrease the number of states at all.

Theorem 4.3 [AF98] *Any 1-way probabilistic finite automaton recognizing L_p with probability $1/2 + \epsilon$, for a fixed $\epsilon > 0$, has at least p states.*

Proof. Assume that there is a 1-way probabilistic finite automaton with less than p states recognizing L_p with probability $\frac{1}{2} + \epsilon$, for a fixed $\epsilon > 0$. Since the language L_p is in a single-letter alphabet, the automaton can be described as a Markov chain. We use the classification of Markov chains described in Section 2 of [KS60] . According to this classification, the states of the Markov chain (the automaton) are divided into ergodic and transient states. An ergodic set of states is a set which cannot be left once it is entered. A transient set of states is a set in which every state can be reached from every other state, and which can be left. An ergodic state is an element of an ergodic set. A transient state is an element of a transient set. If a Markov chain has more than one ergodic set, then there is absolutely no interaction between these sets. Hence we have two or more unrelated Markov chains lumped together. These chains may be studied separately. If a Markov chain consists of a single ergodic set, the chain is called an ergodic chain.

According to results in Section 2 of [KS60], every ergodic chain is either regular or cyclic.

If a Markov chain is regular, then sufficiently high powers of the state transition matrix P of the Markov chain are with all positive elements. Thus no matter where the process starts, after sufficient lapse of time it can be in any state. Moreover, by Theorem 4.2.1 of [KS60] there is limiting vector of probabilities of being in the states of the chain, not dependent of the initial state.

If a Markov chain is cyclic, then the chain has a period d, and its states are subdivided into d cyclic sets ($d > 1$). For a given starting position, it moves through the cyclic sets in a definite order, returning to the set of the starting state after d steps. Hence the d-th power of the state transition matrix P describes a regular Markov chain.

We have assumed that p is prime, and the automaton has less than p states. Hence for every cyclic state of the automaton the value of d is strictly less than p, and because of primality of p, d is relatively prime to p. By D we denote the least common multiple of all such values d. Hence D is relatively prime to p, and so is any positive degree D^n of D. Since $1^{D^n} \notin M_p$ but $1^{D^n p} \in M_p$, the total of the probabilities to be in an accepting state exceeds $\frac{1}{2} + \epsilon$ for 1^{D^n} and is less than $\frac{1}{2} - \epsilon$ for $1^{D^n p}$. Contradiction with Theorem 4.2.1 of [KS60]. □

Corollary 4.1 [AF98] *For the language L_p, the number of states needed by a classical (deterministic or probabilistic) 1-way automaton is exponential in the number of states of a 1-way QFA.*

However the reader should not make a conclusion that quantum finite automata are always more succinct than the deterministic ones. We already cited above Theorem 3.1 by Kondacs and Watrous showing that some regular languages cannot be recognized by quantum finite automata with any probability bounded away from $\frac{1}{2}$. A. Ambainis, A. Nayak, A. Ta-Shma, and U. Vazirani [ANTV99] showed additionally that there is a language which can be recognized by quantum finite automata but QFAs are exponentially worse in size.

Theorem 4.4 [ANTV99] *Consider the finite regular language* $L_n = \{w0|w \in \{0,1\}^*, |w| \leq n\}$.

1. *L_n is accepted by a deterministic finite automaton of size $O(n)$,*

2. *any quantum finite automaton recognizing L_n with a probability bounded away from $\frac{1}{2}$ has a size $2^{\Omega(n/\log n)}$.*

Later Ashvin Nayak [Na99] improved the lower bound for the same language to $2^{\Omega(n)}$ by using technique derived from A. Holevo's theorem [Ho73].

For probabilistic computation, the property that the probability of correct answer can be increased arbitrarily is considered evident. Hence, it is not surprising that [KW97] wrote "with error probability bounded away from $1/2$", thinking that all such probabilities are equivalent. However, mixing reversible (quantum computation) and non-reversible (measurements after each step) components in one model makes it impossible. This problem was first considered by A. Ambainis and R. Freivalds[AF98]. The following theorems were proved there:

Theorem 4.5 [AF98]. *Let L be a language and M be its minimal automaton. Assume that there is a word x such that M contains states q_1, q_2 satisfying:*

1. *$q_1 \neq q_2$,*
2. *If M starts in the state q_1 and reads x, it passes to q_2,*
3. *If M starts in the state q_2 and reads x, it passes to q_2, and*
4. *q_2 is neither "all-accepting" state, nor "all-rejecting" state.*

Then L cannot be recognized by a 1-way quantum finite automaton with probability $7/9 + \epsilon$ for any fixed $\epsilon > 0$.

Theorem 4.6 [AF98]. *Let L be a language and M be its minimal automaton. If there is no q_1, q_2, x satisfying conditions of Theorem 4.5 then L can be recognized by a 1-way reversible finite automaton (i.e. L can be recognized by a 1-way quantum finite automaton with probability 1).*

Theorem 4.7 [AF98]. *The language a^*b^* can be recognized by a 1-way QFA with the probability of correct answer $p = 0.68...$ where p is the root of $p^3 + p = 1$.*

Corollary 4.2 [AF98]. *The language a^*b^* can be recognized by a 1-QFA with probability $0.68...$ but not with probability $7/9 + \epsilon$.*

In the paper [ABFK99] Ambainis, Bonner, Freivalds and Ķikusts consider the best probabilities of acceptance by 1-way quantum finite automata the languages $a^*b^*\ldots z^*$. Since the reason why the language a^*b^* cannot be accepted by 1-way quantum finite automata is the property described in the Theorems 4.5 and 4.6, this new result provides an insight on what the hierarchy of languages with respect to the probabilities of their acceptance by 1-way quantum finite automata may be.

5 A hierarchy of languages

Lemma 5.1 [ABFK99]. *For arbitrary real $x_1 > 0$, $x_2 > 0$, ..., $x_n > 0$, there exists such a unitary $n \times n$ matrix $M_n(x_1, x_2, ..., x_n)$ with elements m_{ij} that*

$$m_{11} = \frac{x_1}{\sqrt{x_1^2 + ... + x_n^2}},\ m_{12} = \frac{x_2}{\sqrt{x_1^2 + ... + x_n^2}},\ ..., m_{1n} = \frac{x_n}{\sqrt{x_1^2 + ... + x_n^2}}.$$

□

Let L_n be the language $a_1^*a_2^*...a_n^*$.

Theorem 5.1 [ABFK99]. *The language L_n ($n > 1$) can be recognized by a 1-way QFA with the probability of correct answer p where p is the root of $p^{\frac{n+1}{n-1}} + p = 1$ in the interval $[\frac{1}{2}, 1]$.*

Proof Let m_{ij} be the elements of matrix $M_k(x_1, x_2, ..., x_k)$ from Lemma 5.1. We construct a $k \times (k-1)$ matrix $T_k(x_1, x_2, ..., x_k)$ with elements $t_{ij} = m_{j+1,i}$. Let $R_k(x_1, x_2, ..., x_k)$ be a $k \times k$ matrix with elements $r_{ij} = \frac{x_i \cdot x_j}{x_1^2 + ... + x_k^2}$ and I_k be the $k \times k$ identity matrix.

For fixed n, let $p_n \in [0, 1]$ satisfy $p_n^{\frac{n+1}{n-1}} + p_n = 1$ and p_k $(1 \leq k < n) = p_n^{\frac{k-1}{n-1}} - p_n^{\frac{k}{n-1}}$. It is easy to see that $p_1 + p_2 + ... + p_n = 1$ and

$$1 - \frac{p_n(p_k + ... + p_n)^2}{(p_{k-1} + ... + p_n)^2} = 1 - \frac{p_n p_n^{\frac{2(k-1)}{n-1}}}{p_n^{\frac{2(k-2)}{n-1}}} = 1 - p_n^{\frac{n+1}{n-1}} = p_n. \qquad (1)$$

Now we describe a 1-way QFA accepting the language L_n.

The automaton has $2n$ states: q_1, q_2, ... q_n are non halting states, q_{n+1}, q_{n+2}, ... q_{2n-1} are rejecting states and q_{2n} is an accepting state. The transition function is defined by unitary block matrices

$$V_\kappa = \begin{pmatrix} M_n(\sqrt{p_1}, \sqrt{p_2}, ..., \sqrt{p_n}) & \mathbf{0} \\ \mathbf{0} & I_n \end{pmatrix},$$

$$V_{a_1} = \begin{pmatrix} R_n(\sqrt{p_1}, \sqrt{p_2}, ..., \sqrt{p_n}) & T_n(\sqrt{p_1}, \sqrt{p_2}, ..., \sqrt{p_n}) & \mathbf{0} \\ T_n^T(\sqrt{p_1}, \sqrt{p_2}, ..., \sqrt{p_n}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix},$$

$$V_{a_2} = \begin{pmatrix} 0 & \mathbf{0} & 1 & \mathbf{0} & 0 \\ \mathbf{0} & R_{n-1}(\sqrt{p_2}, ..., \sqrt{p_n}) & \mathbf{0} & T_{n-1}(\sqrt{p_2}, ..., \sqrt{p_n}) & \mathbf{0} \\ 1 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & T_{n-1}^T(\sqrt{p_2}, ..., \sqrt{p_n}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & \mathbf{0} & 0 & \mathbf{0} & 1 \end{pmatrix},$$

$$...,$$

$$V_{a_k} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & I_{n-k} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & R_k(\sqrt{p_{n+k-1}}, ..., \sqrt{p_n}) & \mathbf{0} & T_k(\sqrt{p_{n+k-1}}, ..., \sqrt{p_n}) & \mathbf{0} \\ I_{n-k} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & T_k^T(\sqrt{p_{n+k-1}}, ..., \sqrt{p_n}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 \end{pmatrix},$$

$$...,$$

$$V_{a_n} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & I_{n-1} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ I_{n-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 \end{pmatrix},$$

$$V_{\$} = \begin{pmatrix} \mathbf{0} & I_n \\ I_n & \mathbf{0} \end{pmatrix}.$$

Case 1. The input is $\kappa a_1^* a_2^* ... a_n^* \$$.

The starting superposition is $|q_1\rangle$. After reading the left endmarker the superposition becomes $\sqrt{p_1}|q_1\rangle + \sqrt{p_2}|q_2\rangle + ... + \sqrt{p_n}|q_n\rangle$ and after reading a_1^* the superposition remains the same.

If the input contains a_k then reading the first a_k changes the non-halting part of the superposition to $\sqrt{p_k}|q_k\rangle + ... + \sqrt{p_n}|q_n\rangle$ and after reading all the rest of a_k the non-halting part of the superposition remains the same.

Reading the right endmarker maps $|q_n\rangle$ to $|q_{2n}\rangle$ Therefore, the superposition after reading it contains $\sqrt{p_n}|q_{2n}\rangle$. This means that the automaton accepts with probability p_n because q_{2n} is an accepting state.

Case 2. The input is $\kappa a_1^* a_2^* ... a_k^* a_k a_m ...$ $(k > m)$.

After reading the last a_k the non-halting part of the superposition is $\sqrt{p_k}|q_k\rangle + ... + \sqrt{p_n}|q_n\rangle$ Then reading a_m changes the non-halting part to

$\frac{\sqrt{p_1}(p_k+\ldots+p_n)}{(p_m+\ldots+p_n)}|q_1\rangle+\ldots+\frac{\sqrt{p_n}(p_k+\ldots+p_n)}{(p_m+\ldots+p_n)}|q_n\rangle$. This means that the automaton accepts with probability $\leq \frac{p_n(p_k+\ldots+p_n)^2}{(p_m+\ldots+p_n)^2}$ and rejects with probability at least

$$1-\frac{p_n(p_k+\ldots+p_n)^2}{(p_m+\ldots+p_n)^2} \geq 1-\frac{p_n(p_k+\ldots+p_n)^2}{(p_{k-1}+\ldots+p_n)^2} = p_n$$

that follows from (1). □

Corollary 5.1 [ABFK99]. *The language L_n can be recognized by a 1-way QFA with the probability of correct answer at least $\frac{1}{2}+\frac{c}{n}$, for a constant c.*

Proof: By resolving the equation $p^{\frac{n+1}{n-1}}+p=1$, we get $p=\frac{1}{2}+\Theta(\frac{1}{n})$. □

Theorem 5.2 [ABFK99]. *The language L_n cannot be recognized by a 1-way QFA with probability greater than p where p is the root of*

$$(2p-1)=\frac{2(1-p)}{n-1}+4\sqrt{\frac{2(1-p)}{n-1}} \tag{2}$$

in the interval $[\frac{1}{2},1]$.

Proof: Assume we are given a 1-way QFA M. We will show that, for any $\epsilon>0$, there is a word such that the probability of correct answer is less than $p+\epsilon$.

Lemma 5.2 [AF98] *Let $x\in\Sigma^+$. There are subspaces E_1, E_2 such that $E_{non}=E_1\oplus E_2$ and*

(i) If $\psi\in E_1$, then $V_x(\psi)\in E_1$,

(ii) If $\psi\in E_2$, then $\|V'_{x^k}(\psi)\|\to 0$ when $k\to\infty$.

We will use $n-1$ decompositions of this type: with $x=a_2$, $x=a_3$, ..., $x=a_{n-1}$ and $x=a_n$. The subspaces E_1, E_2 corresponding to $x=a_m$ will be denoted $E_{m,1}$ and $E_{m,2}$.

Let $m\in\{2,\ldots,n\}$, $y\in a_1^*a_2^*\ldots a_{m-1}^*$. ψ_y denotes the superposition after reading y (with observations w.r.t. $E_{non}\oplus E_{acc}\oplus E_{rej}$ after every step). We express it as $\psi_y^1+\psi_y^2$, $\psi_y^1\in E_{m,1}$, $\psi_y^2\in E_{m,2}$.

Case 1. There is $m\in\{2,\ldots,n\}$ and $y\in a_1^*\ldots a_{m-1}^*$ such that $\|\psi_y^2\|\leq\sqrt{\frac{2(1-p)}{n-1}}$.

We consider inputs ya_{m-1} and $ya_m^i a_{m-1}$, for an appropriate $i > 0$. Then, $ya_{m-1} \in L_n$ but $ya_m^i a_{m-1} \notin L_n$. We consider the distributions of probabilities on M's answers "accept" and "reject" on ya_{m-1} and $ya_m^i a_{m-1}$. If M recognizes L_n with probability $p+\epsilon$, it must accept ya_{m-1} with probability at least $p+\epsilon$ and reject it with probability at most $1-p-\epsilon$. Also, $ya_m^i a_{m-1}$ must be rejected with probability at least $p+\epsilon$ and accepted with probability at most $1-p-\epsilon$. Therefore, both the probabilities of accepting and the probabilities of rejecting must differ by at least

$$(p+\epsilon) - (1-p-\epsilon) = 2p - 1 + 2\epsilon.$$

This means that the *variational distance* between two probability distributions (the sum of these two distances) must be at least $2(2p-1)+4\epsilon$. We show that it cannot be so large.

First, we select i. Let k be so large that $\|V'_{a_m^k}(\psi_y^2)\| \leq \delta$ for $\delta = \epsilon/4$. $\psi_y^1, V'_{a_m}(\psi_y^1), V'_{a_m^2}(\psi_y^1), \ldots$ is a bounded sequence in a finite-dimensional space. Therefore, it has a limit point and there are i, j such that

$$\|V'_{a_m^j}(\psi_y^1) - V'_{a_m^{i+j}}(\psi_y^1)\| < \delta.$$

We choose i, j so that $i > k$.

The difference between two probability distributions comes from two sources. The first source is difference between ψ_y and $\psi_{ya_m^i}$ (the states of M before reading a_{m-1}). The second source is the possibility of M accepting while reading a_m^i (the only part that is different in the two words). We bound each of them.

The difference $\psi_y - \psi_{ya_m^i}$ can be partitioned into three parts.

$$\psi_y - \psi_{ya_m^i} = (\psi_y - \psi_y^1) + (\psi_y^1 - V'_{a_m^i}(\psi_y^1)) + (V'_{a_m^i}(\psi_y^1) - \psi_{ya_m^i}). \qquad (3)$$

The first part is $\psi_y - \psi_y^1 = \psi_y^2$ and $\|\psi_y^2\| \leq \sqrt{\frac{2(1-p)}{n-1}}$. The second and the third parts are both small. For the second part, notice that V'_{a_m} is unitary on $E_{m,1}$ (because V_{a_m} is unitary and $V_{a_m}(\psi)$ does not contain halting components for $\psi \in E_{m,1}$). Hence, V'_{a_m} preserves distances on $E_{m,1}$ and

$$\|\psi_y^1 - V'_{a_m^i}(\psi_y^1)\| = \|V'_{a_m^j}(\psi_y^1) - V'_{a_m^{i+j}}(\psi_y^1)\| < \delta$$

For the third part of (3), remember that $\psi_{ya_m^i} = V'_{a_m^i}(\psi_y)$. Therefore,

$$\psi_{ya_m^i} - V'_{a_m^i}(\psi_y^1) = V'_{a_m^i}(\psi_y) - V'_{a_m^i}(\psi_y^1) = V'_{a_m^i}(\psi_y - \psi_y^1) = V'_{a_m^i}(\psi_y^2)$$

and $\|\psi^2_{ya^i_m}\| \leq \delta$ because $i > k$. Putting all three parts together, we get

$$\|\psi_y - \psi_{ya^i_m}\| \leq \|\psi_y - \psi^1_y\| + \|\psi^1_y - \psi^1_{ya^i_m}\| + \|\psi^1_{ya^i_m} - \psi_{ya^i_m}\| \leq \sqrt{\frac{2(1-p)}{n-1}} + 2\delta.$$

Next, we apply a lemma from [BV97].

Lemma 5.3 [BV97] *Let ψ and ϕ be such that $\|\psi\| \leq 1$, $\|\phi\| \leq 1$ and $\|\psi - \phi\| \leq \epsilon$. Then the total variational distance resulting from measurements of ϕ and ψ is at most 4ϵ.*

This means that the difference between any probability distributions generated by ψ_y and $\psi_{ya^i_m}$ is at most

$$4\sqrt{\frac{2(1-p)}{n-1}} + 8\delta.$$

In particular, this is true for the probability distributions obtained by applying $V_{a_{m-1}}$, $V_\$$ and the corresponding measurements to ψ_y and $\psi_{ya^i_m}$.

The probability of M halting while reading a^i_m is at most $\|\psi^2_\kappa\|^2 = \frac{2(1-p)}{n-1}$. Adding it increases the variational distance by at most $\frac{2(1-p)}{n-1}$. Hence, the total variational distance is at most

$$\frac{2(1-p)}{n-1} + 4\sqrt{\frac{2(1-p)}{n-1}} + 8\delta = \frac{2(1-p)}{n-1} + 4\sqrt{\frac{2(1-p)}{n-1}} + 2\epsilon.$$

By definition of p, this is the same as $(2p-1) + 2\epsilon$. However, if M distinguishes y and ya^i_m correctly, the variational distance must be at least $(2p-1) + 4\epsilon$. Hence, M does not recognize one of these words correctly.

Case 2. $\|\psi^2_y\| > \sqrt{\frac{2(1-p)}{n-1}}$ for every $m \in \{2, \ldots, n\}$ and $y \in a^*_1 \ldots a^*_{m-1}$.

We define a sequence of words $y_1, y_2, \ldots, y_m \in a^*_1 \ldots a^*_n$. Let $y_1 = a_1$ and $y_k = y_{k-1}a^{i_k}_k$ for $k \in \{2, \ldots, n\}$ where i_k is such that

$$\|V'_{a^{i_k}_k}(\psi^2_{y_{k-1}})\| \leq \sqrt{\frac{\epsilon}{3(n-1)}}.$$

The existence of i_k is guaranteed by (ii) of Lemma 5.2.

We consider the probability that M halts on $y_n = a_1 a^{i_2}_2 a^{i_3}_3 \ldots a^{i_n}_n$ before seeing the right endmarker. Let $k \in \{2, \ldots, n\}$. The probability of M halting while reading the $a^{i_k}_k$ part of y_n is at least

$$\|\psi^2_{y_{k-1}}\|^2 - \|V'_{a^{i_k}_k}(\psi^2_{y_{k-1}})\|^2 > \frac{2(1-p)}{n-1} - \frac{\epsilon}{n-1}.$$

By summing over all $k \in \{2, \ldots, n\}$, the probability that M halts on y_n at least

$$(n-1)\left(\frac{2(1-p)}{n-1} - \frac{\epsilon}{n-1}\right) = 2(1-p) - \epsilon.$$

This is the sum of the probability of accepting and the probability of rejecting. Hence, one of these two probabilities must be at least $(1-p) - \epsilon/2$. Then, the probability of the opposite answer on any extension of y_n is at most $1 - (1 - p - \epsilon/2) = p + \epsilon/2$. However, y_n has both extensions that are in L_n and extensions that are not. Hence, one of them is not recognized with probability $p + \epsilon$. □

By solving the equation (2), we get

Corollary 5.2 [ABFK99]. *L_n cannot be recognized with probability greater than $\frac{1}{2} + \frac{3}{\sqrt{n-1}}$.*

Let $n_1 = 2$ and $n_k = \frac{9n_{k-1}^2}{c^2} + 1$ for $k > 1$ (where c is the constant from Theorem 5.1). Also, define $p_k = \frac{1}{2} + \frac{c}{n_k}$. Then, Corollaries 5.1 and 5.2 imply

Theorem 5.3 [ABFK99]. *For every $k > 1$, L_{n_k} can be recognized with by a 1-way QFA with the probability of correct answer p_k but cannot be recognized with the probability of correct answer p_{k-1}.*

Thus, we have constructed a sequence of languages L_{n_1}, L_{n_2}, ... such that, for each L_{n_k}, the probability with which L_{n_k} can be recognized by a 1-way QFA is smaller than for $L_{n_{k-1}}$.

6 Tight bounds for the probability

It was proved in Corollary 4.2 that the language a^*b^* can be recognized by a 1-QFA with probability 0.68... but not with probability $7/9 + \epsilon$. However there is some difference between the probabilities 0.68... and $7/9 + \epsilon$. Which is correct probability for the recognition of the language a^*b^* ? Recently, A. Ambainis has found the solution [Amb99].

Theorem 6.1 [Amb99]. *The language a^*b^* can be recognized by a 1-way QFA with the probability of correct answer*

$$\frac{52 + 4\sqrt{7}}{81}$$

and no better.

7 Which languages are recognizable by QFA

7.1 Necessary condition

First, we give the new condition which implies that the language is not recognizable by a QFA. Similarly to the previous condition (Theorem 3.2), it can be formulated as a condition about the minimal deterministic automaton of a language.

Theorem 7.1 [AKV00] *Let L be a language. Assume that there are words x, y, z_1, z_2 such that its minimal automaton M contains states q_1, q_2, q_3 satisfying:*

1. $q_2 \neq q_3$,

2. if M starts in the state q_1 and reads x, it passes to q_2,

3. if M starts in the state q_2 and reads x, it passes to q_2,

4. if M starts in the state q_1 and reads y, it passes to q_3,

5. if M starts in the state q_3 and reads y, it passes to q_3,

6. for all words $t \in (x|y)^*$ there exists a word $t_1 \in (x|y)^*$ such that if M starts in the state q_2 and reads tt_1, it passes to q_2,

7. for all words $t \in (x|y)^*$ there exists a word $t_1 \in (x|y)^*$ such that if M starts in the state q_3 and reads tt_1, it passes to q_3,

8. if M starts in the state q_2 and reads z_1, it passes to an accepting state,

9. if M starts in the state q_2 and reads z_2, it passes to a rejecting state,

10. if M starts in the state q_3 and reads z_1, it passes to a rejecting state,

11. if M starts in the state q_3 and reads z_2, it passes to an accepting state.

Then L cannot be recognized by a 1-way QFA.

Proof. We use a lemma from [BV97].

Lemma 7.1 [AKV00] *If ψ and ϕ are two quantum states and $\|\psi - \phi\| < \varepsilon$ then the total variational distance between the probability distributions generated by the same measurement on ψ and ϕ is at most*[1] *2ε.*

We also use a lemma from [AF98].

Lemma 7.2 [AF98] *Let $x \in \Sigma^+$. There are subspaces E_1, E_2 such that $E_{non} = E_1 \oplus E_2$ and*

(i) If $\psi \in E_1$, then $V'_x(\psi) \in E_1$ and $\|V'_x(\psi)\| = \|\psi\|$,

[1] The lemma in [BV97] has 4ε but it can be improved to 2ε.

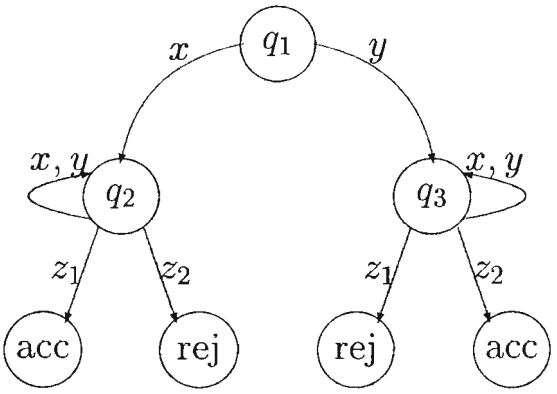


Figure 2: Conditions of theorem 7.1, conditions 6 and 7 are shown symbolically

(ii) If $\psi \in E_2$, *then* $\|V'_{x^k}(\psi)\| \to 0$ *when* $k \to \infty$.

Lemma 7.2 can be viewed as a quantum counterpart of the *classification of states for Markov chains* [KS60]. The classification of states divides the states of a Markov chain into *ergodic* sets and *transient* sets. If the Markov chain is in an ergodic set, it never leaves it. If it is in a transient set, it leaves it with probability $1 - \epsilon$ for an arbitrary $\epsilon > 0$ after sufficiently many steps.

In the quantum case, E_1 is the counterpart of an ergodic set: if the quantum random process defined by repeated reading of x is in a state $\psi \in E_1$, it stays in E_1. E_2 is a counterpart of a transient set: if the state is $\psi \in E_2$, E_2 is left (for an accepting or rejecting state) with probability arbitrarily close to 1 after sufficiently many x's.

The next Lemma is our generalization of Lemma 7.2 for the case of two different words x and y.

Lemma 7.3 [AKV00] *Let* $x, y \in \Sigma^+$. *There are subspaces* E_1, E_2 *such that* $E_{non} = E_1 \oplus E_2$ *and*

(i) If $\psi \in E_1$, *then* $V'_x(\psi) \in E_1$ *and* $V'_y(\psi) \in E_1$ *and* $\|V'_x(\psi)\| = \|\psi\|$ *and* $\|V'_y(\psi)\| = \|\psi\|$,

(ii) If $\psi \in E_2$, *then for any* $\epsilon > 0$, *there exists a word* $t \in (x|y)^*$ *such that* $\|V'_t(\psi)\| < \epsilon$.

Proof. We use E_1^z to denote the space E_1 from Lemma 7.2 for a word z. We define $E_1 = \bigcap_{z \in (x|y)^*} E_1^z$. E_2 consists of all vectors in E_{non} orthogonal to

E_1. Next, we check that both (i) and (ii) are true.

(i) It is easy to see that, for all $t \in (x|y)^*$, $\|V'_t(\psi)\| = \|\psi\|$ due to $\psi \in E_1^t$.

We also need to prove that $V'_x(\psi) \in E_1$ and $V'_y(\psi) \in E_1$. For a contradiction, assume there are $\psi \in E_1$ and $t_1 \in (x|y)^*$ such that $V'_{t_1}(\psi) \notin E_1$. Then, by definition of E_1, there also exists $t_2 \in (x|y)^*$ such that $V'_{t_1}(\psi)$ does not belong to $E_1^{t_2}$. Lemma 7.2 implies that the norm of $V'_{t_1}(\psi)$ can be decreased by repeated applications of V'_{t_2}. A contradiction with $\|V'_t(\psi)\| = \|\psi\|$ for all t.

(ii) Clearly, if ψ belongs to E_2 then for all $t \in (x|y)^*$ the superposition $V'_t(\psi)$ also belongs to E_2 because V_x and V_y are unitary and map E_1 to itself (and, therefore, any vector orthogonal to E_1 is mapped to a vector orthogonal to E_1).

$\|V'_t(\psi)\|$ does not increase if we extend the word t to the right and it is bounded from below by 0. Hence, for any fixed ϵ we can find a $t \in (x|y)^*$ such that

$$\|V'_t(\psi)\| - \|V'_{tw}(\psi)\| < \epsilon$$

for all $w \in (x|y)^*$. We define a sequence of such words $t_1, t_2, t_3, \ldots$ for $\epsilon, \frac{\epsilon}{2}, \frac{\epsilon}{4}, \ldots$. $V'_{t_1}(\psi), V'_{t_2}(\psi), V'_{t_3}(\psi), \ldots$ is a bounded sequence in a finite dimensional space. Therefore, it has a limit point ψ'. We will show that $\psi' = 0$.

First, notice that $\psi' \in E_2$ because it is a limit of a subsequence of $V'_{t_1}(\psi), V'_{t_2}(\psi), V'_{t_3}(\psi), \ldots$ and all $V'_{t_i}(\psi)$ belong to E_2. Therefore, if $\psi' \neq 0$ then ψ' has nonzero E_2^z component for some $z \in (x|y)^*$. Reading sufficiently many z would decrease this component, decreasing the norm of ψ'.

This contradicts the fact that, for any w, $\|\psi'\| = \|V'_w(\psi')\|$ (since $\|\psi'\| - \|V'_w(\psi')\|$ is less than any $\epsilon > 0$ which is true because ψ' is the limit of $V'_{t_1}(\psi), V'_{t_2}(\psi), V'_{t_3}(\psi), \ldots$).

Therefore, $\psi' = 0$. This completes the proof of lemma. □

Let L be a language such that its minimal automaton M contains the "forbidden construction" and M_q be a QFA. We show that M_q does not recognize L.

Let w be a word after reading which M is in the state q_1. Let $\psi_w = \psi_w^1 + \psi_w^2$, $\psi_w^1 \in E_1$, $\psi_w^2 \in E_2$. We find a word $a \in (x|y)^*$ such that after reading xa M is in the state q_2 and the norm of $\psi_{wxa}^2 = V'_a(\psi_{wx}^2)$ is at most some fixed $\epsilon > 0$. (Such word exists due to Lemma 7.3 and conditions 6 and 7.) We also find a word b such that $\|\psi_{wyb}^2\| \leq \epsilon$.

Because of unitarity of V'_x and V'_y on E_1 (part (i) of Lemma 7.3), there exist integers i and j such that $\|\psi^1_{w(xa)^i} - \psi^1_w\| \leq \epsilon$ and $\|\psi^1_{w(yb)^j} - \psi^1_w\| \leq \epsilon$.

Let p be the probability of M_q accepting while reading κw. Let p_1 be the probability of accepting while reading $(xa)^i$ with a starting state ψ_w, p_2 be the probability of accepting while reading $(yb)^j$ with a starting state ψ_w and p_3, p_4 be the probabilities of accepting while reading $z_1\$$ and $z_2\$$ with a starting state ψ^1_w.

Let us consider four words

$$\kappa w(xa)^i z_1\$,$$
$$\kappa w(xa)^i z_2\$,$$
$$\kappa w(yb)^j z_1\$,$$
$$\kappa w(yb)^j z_2\$.$$

Lemma 7.4 [AKV00] *M_q accepts $\kappa w(xa)^i z_1\$$ with probability at least $p + p_1 + p_3 - 4\epsilon$ and at most $p + p_1 + p_3 + 4\epsilon$.*

Proof. The probability of accepting while reading κw is p. After that, M_q is in the state ψ_w and reading $(xa)^i$ in this state causes it to accept with probability p_1.

The remaining state is $\psi_{w(xa)^i} = \psi^1_{w(xa)^i} + \psi^2_{w(xa)^i}$. If it was ψ^1_w, the probability of accepting while reading the rest of the word ($z_1\$$) would be exactly p_3. It is not quite ψ^1_w but it is close to ψ^1_w. Namely, we have

$$\|\psi_{w(xa)^i} - \psi^1_w\| \leq \|\psi^2_{w(xa)^i}\| + \|\psi^1_{w(xa)^i} - \psi^1_w\| \leq \epsilon + \epsilon = 2\epsilon.$$

By Lemma 7.1, this means that the probability of accepting during $z_1\$$ is between $p_3 - 4\epsilon$ and $p_3 + 4\epsilon$. □

Similarly, on the second word M_q accepts with probability between $p + p_1 + p_4 - 4\epsilon$ and $p + p_1 + p_4 + 4\epsilon$. On the third word M_q accepts with probability between $p + p_2 + p_3 - 4\epsilon$ and $p + p_2 + p_3 + 4\epsilon$. On the fourth word M_q accepts with probability $p + p_2 + p_4 - 4\epsilon$ and $p + p_2 + p_4 + 4\epsilon$.

This means that the sum of accepting probabilities of two words that belong to L (the first and the fourth words) differs from the sum of accepting probabilities of two words that do not belong to L (the second and the third) by at most 16ϵ. Hence, the probability of correct answer of M_q on one of these words is at most $\frac{1}{2} + 4\epsilon$. Since such 4 words can be constructed for arbitrarily small ϵ, this means that M_q does not recognize L. □

7.2 Necessary and sufficient condition

For languages whose minimal automaton does not contain the construction of Figure 3, this condition (together with Theorem 3.2) is necessary and sufficient.

Theorem 7.2 [AKV00] *Let U be the class of languages whose minimal automaton does not contain "two cycles in a row" (Fig. 3). A language that belongs to U can be recognized by a 1-way QFA if and only if its minimal deterministic automaton does not contain the "forbidden construction" from Theorem 3.2 and the "forbidden construction" from Theorem 7.1.*

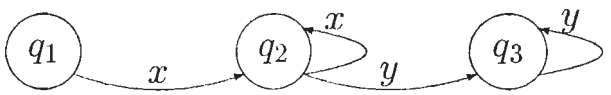


Figure 3: Conditions of theorem 7.2

8 Non-closure under union

8.1 Non-closure result

In particular, Theorem 7.1 implies that the class of languages recognized by QFAs is not closed under union.

Let L_1 be the language consisting of all words that start with any number of letters a and after first letter b (if there is one) there is an odd number of letters a.

This language satisfies the conditions of Theorem 7.1. (q_1, q_2 and q_3 of Theorem 7.1 are just q_1, q_2 and q_3 of G_1. x, y, z_1 and z_2 are b, aba, ab and b.) Hence, it cannot be recognized by a QFA.

Consider 2 other languages L_2 and L_3 defined as follows.

L_2 consists of all words which start with an even number of letters a and after first letter b (if there is one) there is an odd number of letters a.

L_3 consists of all words which start with an odd number of letters a and after first letter b (if there is one) there is an odd number of letters a.

It is easy to see that $L_1 = L_2 \bigcup L_3$.

The minimal automata for these languages do not contain any of the "forbidden constructions" of Theorem 7.2. Therefore, L_2 and L_3 can be recognized by a QFA and we get

Theorem 8.1 [Val00] *There are two languages L_2 and L_3 which are recognizable by a QFA but the union of them $L_1 = L_2 \bigcup L_3$ is not recognizable by a QFA.*

Corollary 8.1 [Val00] *The class of languages recognizable by a QFA is not closed under union.*

This answers a question of Brodsky and Pippenger [BP99].

As $L_2 \bigcap L_3 = \emptyset$ then also $L_1 = L_2 \Delta L_3$. So the class of languages recognizable by QFA is not closed under symmetric difference. ¿From this and from the fact that this class is closed under complement, it easily follows:

Corollary 8.2 [Val00] *The class of languages recognizable by a QFA is not closed under any binary boolean operation where both arguments are significant.*

References

[AF98] Andris Ambainis and Rūsiņš Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. *Proc. 39th FOCS*, 1998, p. 332– 341.
*http : //www.arxiv.org/abs/quant − ph/*9802062

[ABFGK99] Andris Ambainis, Richard Bonner, Rūsiņš Freivalds, Marats Golovkins and Marek Karpinski. Quantum Finite Multitape Automata. *Lecture Notes in Computer Science*, Springer-Verlag, 1999, vol. 1725, p. 340–348.

[ABFK99] Andris Ambainis, Richard Bonner, Rūsiņš Freivalds and Arnolds Ķikusts. Probabilities to accept languages by quantum finite automata. *Lecture Notes in Computer Science*, Springer-Verlag, 1999, vol. 1627, p. 174–183.

[AKV00] Andris Ambainis, Arnolds Ķikusts, Māris Valdats. On the class of languages recognizable by 1-way quantum finite automata.
*http : //www.arxiv.org/abs/quant − ph/*0009004

[ANTV99] Andris Ambainis, Ashvin Nayak, Amnon Ta-Shma, and Umesh Vazirani. Dense quantum coding and a lower bound for 1-way quantum automata. *Proc. STOC'99,* 1999, p. 376–383.
*http : //www.arxiv.org/abs/quant − ph/*9804043

[Amb96] Andris Ambainis. The complexity of probabilistic versus deterministic finite automata. *Proceedings of the International Symposium on Algorithms and Computation(ISAAC'96)*, Lecture Notes in Computer Science, vol. 1178, 1996, p. 233–239.

[Amb99] Andris Ambainis. Personal communication. 1999.

[BBF00] Aija Bērziņa, Richard Bonner and Rūsiņš Freivalds. Parameters in Ambainis-Freivalds algorithm. *Proc. International Workshop "Quantum Computation and Learning", Sundbyholms Slott, Sweden*, 2000, p. 101 109.

[BFG00] Richard Bonner, Rūsiņš Freivalds and Renārs Gailis. Undecidability of 2-tape quantum finite automata. *Proc. International Workshop "Quantum Computation and Learning", Sundbyholms Slott, Sweden*, 2000, p. 93 100.

[BFK00] Richard Bonner, Rūsiņš Freivalds and Maksim Kravtsev. Quantum versus probabilistic 1-way finite automata with counter. *Proc. International Workshop "Quantum Computation and Learning", Sundbyholms Slott, Sweden*, 2000, p. 80–88.

[BP99] A. Brodsky, N. Pippenger. Characterizations of 1-way quantum finite automata.
http://www.arxiv.org/abs/quant-ph/9903014

[BV97] Ethan Bernstein, Umesh Vazirani, Quantum complexity theory. *SIAM Journal on Computing*, vol. 26, 1997, p. 1411–1473.

[Fr79] Rūsiņš Freivalds. Fast probabilistic algorithms. *Lecture Notes in Computer Science*, 1979, vol. 74, p. 57 69.

[Fre82] Rūsiņš Freivalds. On the growth of the number of states in result of determinization of probabilistic finite automata. *Avtomatika i Vychislitel'naya Tehnika*, 1982, no. 3, p. 39–42 (in Russian).

[Gr00] Jozef Gruska. Descriptional complexity issues in quantum computing. *Journal of Automata, Languages and Combinatorics*, vol. 5, 2000, p. 191–218.

[Ho73] Alexander S. Holevo. Some estimates of the information transmitted by quantum communication channels. *Problemy Peredachi Informatsii*, vol. 9, 1973, p. 3–11 (in Russian). English translation: *Problems of Information Transmission*, vol. 9, 1973, p. 177–183.

[KS60] John G. Kemeny and J. Laurie Snell. *Finite Markov Chains.* Van Nostrand, Princeton, NJ, 1960.

[Ki98] Arnolds Ķikusts. A small 1-way quantum finite automaton. *Proc. International Workshop "Quantum Computation and Learning", Riga, Latvia*, 1999, p. 78–83.
*http : //www.arxiv.org/abs/quant − ph/*9810065

[KR00] Arnolds Ķikusts and Zigmārs Rasščevskis. On the accepting probabilities of 1-way quantum finite automata. *Proc. International Workshop "Quantum Computation and Learning", Sundbyholms Slott, Sweden*, 2000, p. 72–79.

[KW97] Attila Kondacs and John Watrous. On the power of quantum finite state automata. In *Proc. 38th FOCS*, 1997, p. 66–75.

[Kra99] Maksim Kravtsev: Quantum Finite One-Counter Automata. *Lecture Notes in Computer Science*, Springer-Verlag, 1999, vol. 1725, p. 431-441.

[MT69] A. Meyer, C. Thompson. Remarks on algebraic decomposition of automata. *Mathematical Systems Theory*, vol. 3, 1969, p. 110–118.

[MC97] Christopher Moore, James P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, to appear.
*http : //www.arxiv.org/abs/quant − ph/*9707031

[Na99] Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. *Proc. FOCS'99*, IEEE, 1999, p. 369–376.
*http : //www.arxiv.org/abs/quant − ph/*9904093

[Sh97] Peter Shor. Polynomial time quantum algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 1997, vol. 26, p. 1484-1509.

[Val00] Māris Valdats. The class of languages recognizable by 1-way quantum finite automata is not closed under union.
*http : //www.arxiv.org/abs/quant − ph/*0001005

[VSBYCC00] Lieven M.K. Vandersypen, Matthias Steffen, Gregory Breyta, Costantino S. Yannoni, Richard Cleve, Isaac L. Chuang. Experimental realization of order-finding with a quantum computer.
*http : //www.arxiv.org/abs/quant − ph/*0007017

Quantum Puzzles, Mysteries and Paradoxes

Jozef Gruska* and Ferdinand Peper

Faculty of Informatics, MasarykUniversity,
Botanická 68a, 602 00 Brno, Czech Republic
KARC, 588 Iwaoka-cho, Nishi-ku, Kobe-city, 651-2492, Japan

(Extended preliminary abstract)

Abstract

This paper discusses various puzzling, mysterious and even paradoxical phenomena in quantum physics that are (or should be) of interest and importance for quantum information processing and also for all areas of science and technology (and also of medicine (?)) in which explorations for principles and fundamentals or advances of technologies start to reach quantum level. All areas of Nature computing as well as all attempts to push silicon technology to its limits should be considered to be of this type. A special position in this regard have attempts to explore the existence and impacts of quantum phenomena when trying to get a deep understanding of information processing potentials and methods of the brain.

1 Introduction

Quantum physics has already for a long time been well known for a variety of strange, puzzling, mysterious and even paradoxical phenomena that withstood numerous attempts to be dealt with by some of the brightest people in science. As a consequence, most of (quantum) physicists got used to live with the understanding that quantum physics is a superb theory every quantum physicist (should) know how to use, but actually nobody understands it fully. Puzzling, mysterious and paradoxical quantum phenomena

*The paper has been written during the first author stay with Kansai Advanced Research Institute near Kobe, in summer 2000. Support of the grants GAČR 201/98/0369, CEZ:J07/98:143300001 and VEGA 1/7654/20 is to be acknowledged.

actually did not seem to disturb most of the working (quantum) physicists too much. Only philosophers of science working on different interpretations could come with some explanations with which usually only very few others fully agreed. The situation is quite well characterized by the observation that philosophers of science can hardly agree even on what the term "interpretation of quantum physics" should mean.

Quantum theory performed its prediction role superbly, but lagged behind concerning its other very basic task. Namely, to find explanations for all fundamental quantum phenomena; why they happen, how they happen, and how much they cost in terms of various physical resources. Quantum measurement is perhaps the best example of such a phenomenon about which we still know too little.

There are several reasons nowadays for the urgent needs to deal much more in depth with various puzzling, mysterious and paradoxical quantum phenomena on both a theoretical and experimental level.

- The enormous potentials quantum information processing and communication (QIPC) seem to offer[1], require that we very carefully explore power, limitations, and laws of quantum physics underlying QIPC, in order to clearly see the real potential and pitfalls of the field. This is hardly possible without clarifying and understanding many of the basic quantum puzzles, mysteries and paradoxes (especially those related to entanglement and measurement).

- Attempts have started to make use of various puzzling quantum phenomena, like entanglement (and (weak) non-locality), for example through teleportation, quantum communication, quantum information processing or quantum cryptography, or of quantum interaction-free measurements (for example for imaging and so on).

- Advances in Nature computing, especially in molecular, DNA, and brain computing as well as the existence of various puzzling phenomena in these areas also require freeing quantum physics as much as possible from puzzles.

In addition to the growing need to get a deep theoretical understanding of various puzzling, mysterious and paradoxical quantum phenomena, it is also of increasing importance to realize without loopholes all those crucial

[1]For an extensive presentation and discussion of QIPC and its potentials see Gruska (1999) and also Gruska (2000) for a more concise treatment of the area and for some insights into its development.

experiments on which our current key beliefs in quantum mechanics and quantum information processing power are based. For example Bell experiments[2] or teleportation experiments.

In this paper we discuss first some classical paradoxes and puzzles: Maxwell's demon, Schrödinger's cat, the EPR paradox and quantum measurement paradox. In the second, main, part we discuss new paradoxes related with quantum superposition, entanglement, counterfactual effects and measurement.

2 CLASSICAL PARADOXES

Let us first discus briefly some (very) old and famous quantum puzzles/paradoxes. For more details see Gruska (1999) and references there.

2.1 Maxwell's demon

It seems that it was for the first time in connection with the attempts to explain Maxwell's demon paradox, from 1867, that information processing considerations started to play a role in physics.

Maxwell's demon is a creation that operates a shutter to open and to close a trapdoor between two compartments A and B of a completely isolated chamber containing a gas of molecules with a random distribution of particles and velocities. The demon pursues the policy of opening the door only when a fast molecule approaches it from the right, or a slow one from the left. Hence the compartment A cools down and the compartment B heats up. Working in this way for a while the demon separates hot molecules from cold and establishes a temperature difference between two compartments and decreases the entropy of the system without doing any work—apparently violating the second law of thermodynamics .

A real explanation came only after a deeper insight into the thermodynamic cost of information processing was obtained by Landauer and Bennett. It is based on the modern understanding that not information acquisition but information erasure requires energy. In a very simplified form the explanation goes as follows.

The demon has to collect and store information in his memory about the locations and speeds of the molecules. Since his memory is finite he has to erase information from his memory from time to time, and it is during

[2]Bell experiments are experiments to test the original Bell inequalities, or one of the many inequalities that follow from locality.

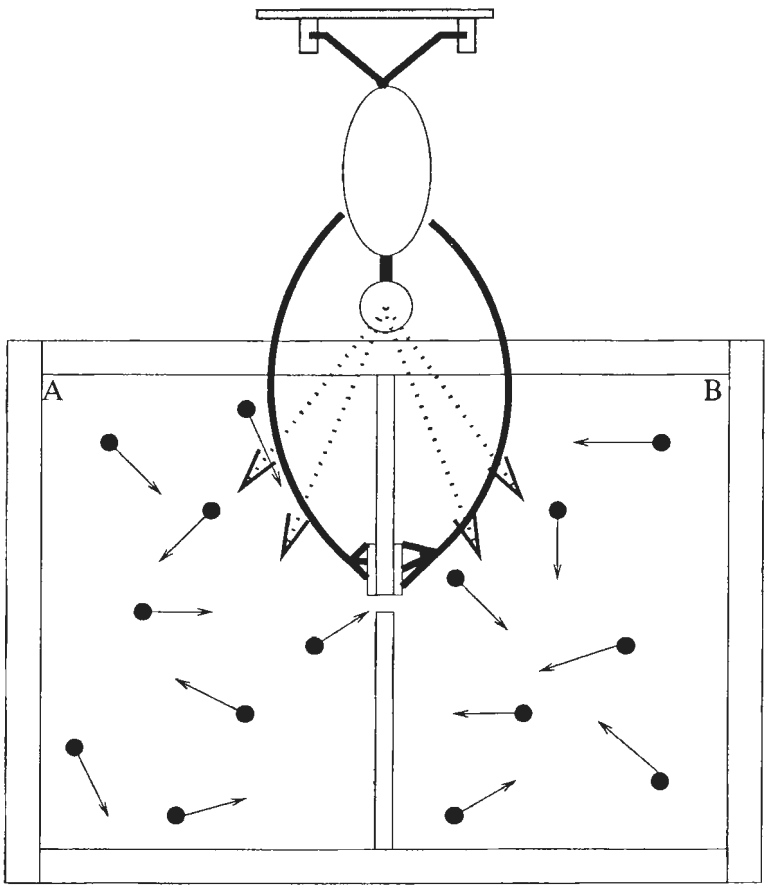


Figure 1: Maxwell's demon

this process that the entropy increases as required by the second law of thermodynamics.

2.2 Schrödinger's cat – quantum superposition puzzle

One of the most puzzling phenomena in our physical world, which is basically quantum mechanical, is why there is no quantum superposition on "macroscopic scale objects" (with the exception of such phenomena as superconductivity). Or, does it actually exist and we are only not able to observe it? We can consider a given macroscopic system also as a microscopic system that develops according to a unitary evolution. There is therefore an apparent contradiction here that was made very vivid through the famous

Gedanken experiment of Schrödinger's cat.

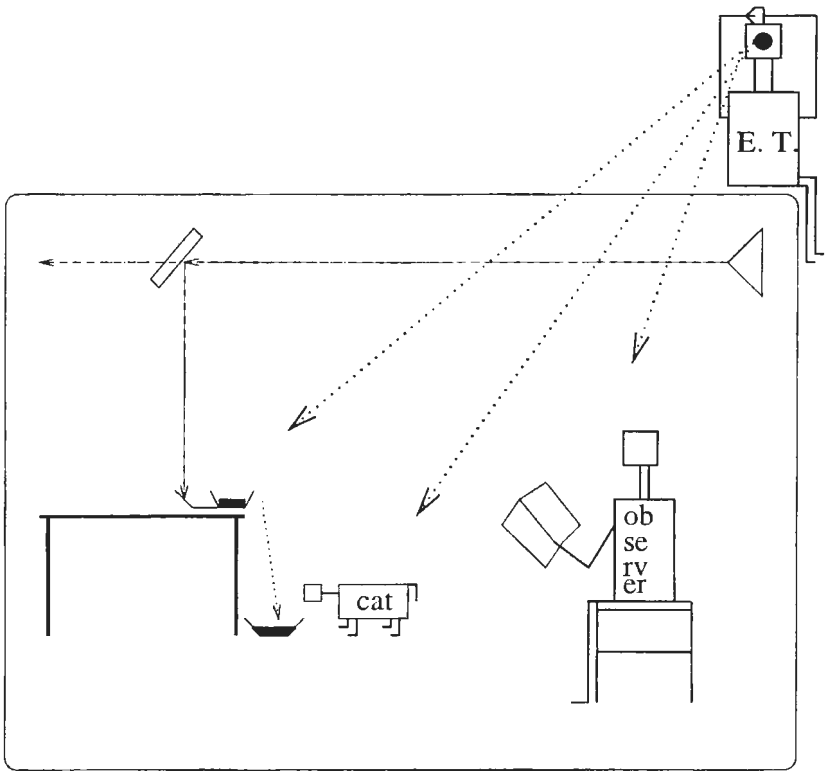


Figure 2: Schrödinger's cat

Let us assume we have a completely isolated chamber with four key elements: an observer, a cat with a cat-food pot, a cup of poison, and an apparatus controlled by a beam of photons from a photon source, also inside the chamber. The beam of photons is directed, as in a Mach-Zehnder interferometer to a half-silvered mirror (see Figure 2). When a photon gets through the mirror, nothing particular happens. The cat keeps having a good time. If the photon is reflected at the mirror it triggers a photo-cell (as a measuring device) and this causes poison to leak from the cup to the cat-pot and the cat dies immediately.

From the point of view of an observer in the chamber there are two possibilities. Either the measuring device, the photo-cell, does not record the photon and the (lucky) cat is alive, or it does and the (poor) cat is dead. There are only these two possibilities and one of them has to happen. For an internal observer there are therefore two options concerning the cat: "alive

or dead", and both have the same probability.

However, the situation looks different to an external observer. He "sees" the whole system in the chamber as a single quantum system in which only unitary evolutions occur, no measurement. From his point of view the photon is in a superposition of two states and the cat gets into the state $\frac{1}{\sqrt{2}}(|\text{alive}\rangle|\uparrow\rangle + |\text{dead}\rangle|\downarrow\rangle)$, that is, she is both alive and dead at the same time but neither of both, with the same probability. However, this contradicts our experience. Cats we see are either alive or dead.

Recently, an understanding of Schrödinger's cat mystery has developed that uses decoherence as the key element and goes briefly as follows. In order to specify fully a quantum state of a cat one needs to specify quantum states of all its components, atoms, electrons,... There is a huge number of quantum states that correspond to alive cats and a huge number of states that correspond to dead cats. All these states constantly evolve due to the inherent interaction of their elements and interactions with the environment. Quantum superposition "both alive and dead but neither of both" can exist but only for an unnoticeable tiny fraction of time, because it is very unstable, and then evolves, due to the decoherence, into a mixed state: alive with a probability one-half and dead with the same probability. In short, Schrödinger's cat does not exist. Or rather it has an immeasurable life-time before it evolves into a classical or Newtonian cat (Lindley, 1996).

2.3 Quantum Measurement

Paradoxes discussed in previous sections are, strictly taken, hardly real paradoxes, because they do not contradict themselves. They discuss puzzling and mysterious phenomena, and require fundamental changes in the way we think about reality, but they are not really paradoxes.

There is, however, one genuine quantum paradox in the Copenhagen interpretation of quantum mechanics, that concerns the key concept of this interpretation--measurement.

On the one hand, quantum physics is considered as the fundamental theory of physics (and therefore, in principle, the workings of all things should be explicable in quantum mechanical terms) and, on the other hand, it assumes that quantum systems exist in undetermined states until they are measured by (classical) measuring devices that have only determined states, even though these devices should also be considered as quantum and therefore being, in principle, also always in undetermined states until their states are measured, ...

Some physicists, including several very prominent ones, even tended to

accept the position that the process of dividing classical parts of measuring devices into classical and quantum parts ends at our consciousness. Namely, that it is not so much the physical act of measurement as the mental act of becoming aware of the result that, finally, creates the borderline between quantum uncertainty and specific knowledge.

A measurement is in the Copenhagen interpretation a step we have to take to get from the quantum world to the classical world, but the Copenhagen interpretation gives hardly an explanation of how this step is performed and how much it costs. It only says what its outcomes are. There does not seem to be anything in quantum theory that explains or determines the exact mechanism of quantum measurements and the resulting state collapses. In particular, both measurements and state collapse are presented in such a way that they would not require involvement of forces of any kind.[3]

Fortunately, the above problem of the quantum measurement paradox seems to be on the way to being solved by means of quantum mechanics itself. Two concepts play a key role by that: decoherence and theoretical insights into the behaviour of complex systems. They allow us to understand how large assembles of weird quantum objects can behave in a reasonable way and how Nature gets around seemingly not understandable quantum measurement phenomena.

2.4 EPR Paradox

Einstein was a strong opponent of the key view of the Copenhagen interpretation of quantum physics, namely that quantum properties are not determined (or that they even do not have a meaning) until they are measured, and insisted that unmeasured quantities must exist in some state even though we might not know what the state is.

According to Einstein, if we can predict with certainty the value of a physical variable without disturbing the system, then there exists an **element of physical reality** that corresponds to that value. In order to discredit views of physics resulting from the Copenhagen interpretation, Albert Einstein, Boris Podolsky and Nathan Rosen (EPR), in their famous paper "Can quantum mechanical description of reality be considered complete?", in 1935, developed a Gedanken experiment that helped to illustrate strange consequences that follow from quantum theory.

[3]Bohr solved "easily" the measurement problem by asserting that measurements can be made and one never had problems to determine whether a prospective measurement is really a measurement, and he just did not bother with the problem of how measurements can be made in general.

Their basic reasoning goes as follows. Let us imagine two particles, whose total momentum is constant, flying apart at the same speed. Once they are far apart you measure the position (or momentum) of the first particle and by that you immediately know the position (momentum) of the second particle.

However, Einstein and his colleagues drew out of that two important conclusions.

1. By measuring precisely the position of the first particle we get precisely the position of the second particle. Since no measurement was involved on the second particle we can now measure its momentum precisely. However, this contradicts Heisenberg's principle.

2. The fact that each measurement of one particle automatically determines the state of another particle is in contradiction with **Einstein's principle of nonlocality**: an action on a physical system A cannot immediately change the state of another space separated system B.

In addition, they developed a viewpoint that if, without disturbing in any way a system (second particle), we can determine with certainty the value of a physical quantity (position or momentum), there has to be an element of physical reality that corresponds to this physical quantity. Therefore, both position and momentum of particles have to be elements of physical reality.

Bohr came up with a surprising, but actually deep explanation why Einstein's reasoning, by which he derived a contradiction with Heisenberg's uncertainty principle, is wrong. According to Bohr one is not allowed to combine into one consideration outcomes of two incompatible measurements (of the position of one particle and of the momentum of the second particle).

An important modification of the basic EPR Gedanken experiment, due to Bohm, shows the problem in an even clearer way. His basic Gedanken experiment dealt with two particles that fly apart in such a way that their spins add up to zero or, in a more modern setting, that they form a pair of entangled qubits, in the total state

$$\frac{1}{\sqrt{2}}(|0\rangle|1\rangle + |1\rangle|0\rangle)$$

and are spatially separated. In such a case a measurement of the spin of one of the qubits determines (observes) uniquely the spin of another qubit without the second one being disturbed by an indirect observation. Einstein called this phenomenon a "spooky-action-at-a-distance" because measurement in one place seems to have an instantaneous effect at the other place.

The term "spooky" indicates that the influence was implied rather than directly seen.

Einstein and his colleagues concluded from their Gedanken experiment that explanations of the real phenomena which current quantum physics offers (namely, its Copenhagen interpretation) are not complete and suggested a way, a program, to fix it—how a proper fundamental theory of Nature should look like. The EPR program asked for *completeness* ("In a complete theory there is an element corresponding to each element of reality."), *locality* ("The real factual situation of system A is independent of what is done with system B, which is spatially separated from the former.") and *reality* ("If, without in any way disturbing a system, we can predict with certainty (i.e., with probability equal to unity) the value of a physical quantity, then there exists an element of physical reality corresponding to this physical quantity.").

3 Quantum Puzzles, Mysteries and Paradoxes for Ever?

Let us now discuss new puzzles, mysteries and paradoxes, actually often only new variations of old ones, but with interesting and important new implications.

3.1 Mysterious entanglement

As already discussed quantum entanglement seems to imply the existence of a (weak) nonlocality feature in Nature. In turn this allows the existence of a variety of puzzling phenomena often termed as telepathy, teleportation and so on. In addition, the way entanglement can be created leads also to puzzling phenomena.

It is usually said that entanglement exhibits only weak nonlocality. By that it is meant that causality can act nonlocally, but without any signaling faster than light.

Let us start our discussion of puzzling phenomena related to entanglement with the presentation of two vividly formulated puzzles that demonstrate that quantum entanglement allows effects resembling telepathy.

3.1.1 Stage telepathy – puzzling behaviour of Alice and Bob.

Alice and Bob are on a stage, far from each other, at the distance of several light years, and they are simultaneously, independently and randomly asked

either a "food question" or a "colour question".

- **FOOD question**: What is your favorite meal?
  ANSWER has to be either **carrot** or **peas**
- **COLOUR question**: What is your favorite colour:
  ANSWER has to be either **green** or **red.**

The audience observes that their answers satisfy the following conditions:

- If both are asked colour-questions then in 9% of the cases they answer **green**
- If one of them is asked colour-question and answers **green** and the other is asked food-question then (s)he answers **peas.**
- If both are asked food-questions they never both answer **peas.**

It is not difficult to show that within the classical physics there is no way that Alice and Bob could coordinate their behaviour in such a way that the above mentioned rules are fulfilled. However, there is a quantum solution, and actually quite an easy one.

Let $|p\rangle$ and $|c\rangle$ be two arbitrary orthogonal states in H_2 and let

$$|r\rangle = a|p\rangle + b|c\rangle$$
$$|g\rangle = b|p\rangle - a|c\rangle$$

Therefore

$$|p\rangle = a|r\rangle + b|g\rangle$$
$$|c\rangle = b|r\rangle - a|g\rangle$$

Let Alice and Bob at the very beginning posses two particles that are in the state

$$|\psi\rangle = N(|r\rangle|r\rangle - a^2|p\rangle|p\rangle)$$

where N is a normalization factor. Then each of them takes his/her particle with him/her.

If any of them is asked the colour-question (s)he measures his/her particle with respect to the $\{|r\rangle, |g\rangle\}$ basis.

If any of them is asked the food-question (s)he measures his/her particle with respect to the $\{|p\rangle, |c\rangle\}$ basis.

It is an easy exercise to derive that in this way Alice's and Bob's responses should follow the rules described above.

3.1.2 Guess my number game — unnatural communication power of Alice, Bob and Charles.

A very nice technical result concerning the communication power of entanglement obtained by van Dam, Hoyer and Tapp (1997) was attractively formulated by Steane and van Dam (2000) as follows.

Alice, Bob and Charles are located in isolated booths and are to guess whether the number of apples, up to 4, that a moderator secretly distributes between them, potentially dividing some apples first into halves and distributing also halves, is even or odd.

The guess is to be made by Alice and the only communication allowed between the contestants is that each of Alice's companions is allowed to sent one bit to Alice (after seeing the portion of apples he got).

Contestants are allowed, before going to isolated booths, to make a strategy and to take with them whatever they want. However, once they are in booths, they are not allowed to communicate in some other way then sending to Alice one bit.

The basic question is whether it is possible that contestants always win? That Alice is always able to determine whether the number of apples distributed is even or odd?

There is no way for Alice, using only classical communications, to guess correctly on more than 75% the correct answer. (Show that!)

Alice would be able to do that would Bob and Charles be allowed to sent her 3 bits. (Show that!)

There is the following quantum solution.

- Before going to booths contestants create three particles in the entangled GHZ state
$$\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$
and each of them takes with him/her one of the particles.

- In case Alice, Bob, and Charles get x_a, x_b or x_c apples, respectively, where $x_a + x_b + x_c$ is an integer smaller than 5, then each of them applies to his/her particle the rotation
$$|0\rangle\langle 0| + e^{ix_i\pi}|1\rangle\langle 1|$$
The resulting state is either
$$\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \text{ if } x_a + x_b + x_c \text{ is even, or}$$

or

$$\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle) \text{ if } x_a + x_b + x_c \text{ is odd.}$$

- In order to enable Alice to determine which of the states they jointly own, each of contestants applies on his/her qubit the Hadamard transformation, getting either the superposition of the basis states of even parity

$$\frac{1}{2}(|000\rangle + |110\rangle + |101\rangle + |011\rangle),$$

or a superposition of states of odd parity

$$\frac{1}{2}(|001\rangle + |010\rangle + |100\rangle + |111\rangle).$$

.

- Each of them measures his/her qubit in the standard basis and Bob and Charles communicate the outcome to Alice.

One can show that in the generalization of the previous game to k contestants

$$\theta(k \lg k) - k$$

bits are needed to be communicated, but if communication enhanced by the entanglement is used, only $k - 1$ bits are needed for communication.

This means that at the entanglement assisted communication one bit is "doing the work" of lgk bits. How is this possible?

One has to realize that what is transmitted is not, strictly speaking, an abstract bit, but rather a two-valued classical signal.

In other words, a real physical entity is transmitted, and it is because this entity is coupled to the entangled system at either end of each transmission that such a gain can be obtained.

An interesting and very important question is whether the above phenomenon of entanglement assisted communication can be demonstrated experimentally; hopefully yes, but this is a very nontrivial task.

3.2 Entanglement swapping

It used to be thought that if some distant particles are in an entangled state, then some time ago they they should have an interaction and only later they have been separated. However, it is now clear that no direct interaction is necessary in order to produce entanglement between distant

quantum systems. For example, entanglement can be teleported. A so-called **entanglement swapping** protocol has been proposed by Zukovski et al. (1993) — and realized by Pan et al. (1998).

The basic idea is simple. If we have two pairs of particles entangled in a Bell state (P_1 with P_2 and P_3 with P_4), Figure 3a, and a Bell measurement on particles P_2 and P_3 is performed, then we get two new entangled pairs: P_1 with P_4 and P_2 with P_3, both in a Bell state. Technically, for any

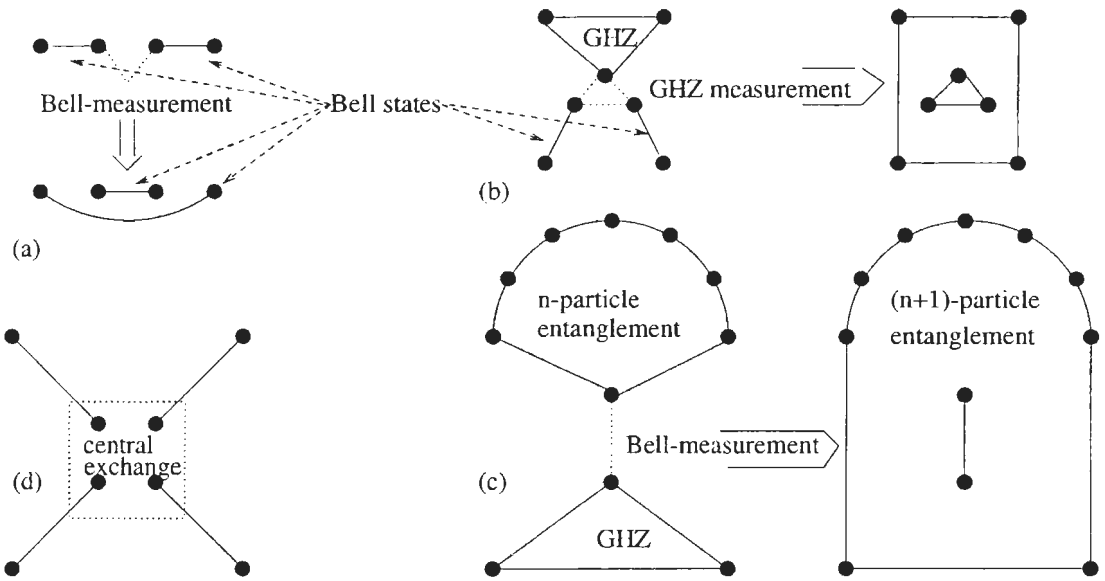


Figure 3: Entanglement swapping and its generalizations

$a, b, c, d \in \{0, 1\}$, if a Bell operator is applied to the state

$$\frac{1}{2}(|abcd\rangle + |\bar{a}\bar{b}cd\rangle + |ab\bar{c}\bar{d}\rangle + |\bar{a}\bar{b}\bar{c}\bar{d}\rangle),$$

to its two middle qubits, then the resulting state is such that the particles P_1 and P_4, as well as P_2 with P_3, are in a Bell state . Observe that entanglement swapping allows superluminal (faster than light) establishment of entanglement between two distant parties.

Entanglement swapping has been experimentally verified by Pan et al. (1998).

A curious modification of entanglement swapping was developed by Peres(2000) using a simple protocol at which entanglement is produced aposteriori, after entangled particles have been measured and may no longer exists (!).

The above procedure has been used to create multiparticle entangled states. In generalized entanglement swapping one deals with generalized n-partite Bell states of the form

$$\frac{1}{\sqrt{2}}(|x\rangle + |\bar{x}\rangle),$$

where $x \in \{0,1\}^n$ and $\bar{x}$ is a bitwise complement of x. Let us assume that we have N sets of entangled particles, each one in a generalized Bell state. Let from the ith set p_i particles be brought together and let a joint Bell measurement be performed on them. (Observe that all sets of Bell states of p particles form an orthogonal basis.) In such a case all measured particles get into a Bell state and the same is true for the rest of the particles.

A generalization of entanglement swapping can also be used to create from an m -partite Bell state and from one GHZ state an $(m+1)$-partite Bell state by simply performing a Bell measurement on one particle of the m-partite state and one particle of the GHZ state, see Figure 3c.

Sometimes flexibility is needed to create "on demand" a generalized Bell state for a group of particles. In such a case the following method can be used, see Figure 3d. Each party involved shares the state $\frac{1}{\sqrt{2}}(|01\rangle - !10\rangle)$ with the center $\mathcal{O}$. If a set of parties wants to share an multipartite state, then the center performs a generalized Bell measurement of its share of singlets to make the corresponding second particles of parties to get into a generalized Bell state.

3.3 Making communication asymptotically more efficient

The above Guess my number puzzle actually exhibits a communication complexity problem for which quantum communication complexity can be asymptotically smaller than classical communication complexity, due to the proper use of entanglement.

So far the asymptotically largest difference between classical and quantum communication complexity, an exponential one, was shown by Raz (1999)..

It is really puzzling that entanglement can have such an enormous influence on communication.

3.4 Entanglement as a catalyst

In some case, for example in the case of teleportation, surprisingly entanglemnt can help to perform communication or transmission of quantum states, but in doing that enangled states get destroyed.

Surprisingly enough, there are also cases when borrowed entanglement can help to achieve what without entanglement cannot be done and at that "used" entangled states do not get destroyed and at the end of the activities entangled states can be "returned".

For example, Jonathan and Plenio (1999) have shown that the mere presence of a (borrowed) entangled state can be an essential advantage when the task is to transform one quantum state into another with local quantum operations and classical communications (LQCC).

They have shown that there are pairs of pure states $(|\phi_1\rangle, |\phi_2\rangle)$ such that using LQCC one cannot transform $|\phi_1\rangle$ into $|\phi_2\rangle$, notation $|\phi_1\rangle \not\to |\phi_2\rangle$(LQCC)), but with the assistance of an appropriate entangled state $|\psi\rangle$ one can transfer $|\phi_1\rangle$ into $|\phi_2\rangle$ using LQCC in such a way that the state $|\psi\rangle$ is not changed in the process (and can be "returned back" after the process) — notation $|\phi_1\rangle \to |\phi_2\rangle$(EQLCC) or $|\psi\rangle \otimes |\phi_1\rangle \not\to |\psi\rangle \otimes |\phi_2\rangle$(LQCC). In such a case $|\psi\rangle$ serves as a "catalyst" for otherwise impossible transformation (reaction).

The above result about the power of entanglement has been generalized by Eisert and Wilkens, as reported at the QIV-Workshop at Bad Honnef, January 10-12, 2000, as follows:

1. There are density matrices (mixed states) σ_1 and σ_2 that correspond to genuinly mixed states such that $\sigma_1 \not\to \sigma_2$(LQCC), but $\sigma_1 \to \sigma_2$(ELQCC), that is, there is an entangled density matrix ρ such that $\rho \otimes \sigma_1 \to \rho \otimes \sigma_2$(ELQCC).

2. The proportion of a pure state in a mixed state can be more efficiently increased with ELQCC operations than using LQCC operations in the following way:

In addition, it is already well known that in the case of borrowed entanglement the capacity of a quantum channel can be smaller than in the case no borrowed entanglement is used.

3.5 Quantum measurement

Reif (1998) pointed out an important problem of quantum computing that so far seems to have received insufficient attention and that needs more exploration to see more clearly perspectives of quantum computing. Namely, as he pointed out, that there appears to be so far neither a mathematical proof nor an experimental demonstration that quantum measurements can be done by measuring devices of small (molecular) volume and with a

small amount of energy. Or, at least, that the amount of these resources needed does not grow exponentially with the number of qubits involved. (In contrary, Reif provides arguments, but not a proof, indicating that exponentially growing resources may be needed to perform even approximate quantum measurements). He has pointed out that the Copenhagen interpretation of the quantum measurement, that assumes that quantum measurements are done by macroscopic devices (of no clear volume, in no clear time, and with no clear amount of energy) does not seem to apply to the case of quantum computing on a molecular level, where one should assume that measurements are done on a microscopic level. In this case the von Neumann interpretation of quantum measurement seems to be more appropriate, at which both the measured state and the measuring device are parts of a quantum system (and measurements are done on the microscopic level first and then outcomes are measured at a macroscopic level to produce classical results) may be more appropriate. It has been pointed out that in such a case the measuring device and the (first) measurement process itself can be subjected to quantum effects and, as a consequence, can lead to predictions different from those the Copenhagen interpretation would provide.

3.6 Bell experiments

Well readable recent analyses of the Bell theorem testing experiments are due to Hardy (1998) and Percival (2000). Here are some main points of the analyses.

- All experiments performed so far make use of untestable supplementary assumptions. For example the "fair sampling assumption". Its basic idea is that "the detectors sample the ensemble in a fair way so that those events in which both photons are detected are representable of the whole ensemble". Or the assumption "that detector efficiency is not dependent on local "hidden variables".

- Without the above assumptions the detectors used in the experiments need to have some efficiency (estimated to be at least 67%) what is far more than what has been achieved so far.

- It is far from clear whether we can performed Bell experiments with ever increasing precision and whether we do not discover some laws of Nature that prevents us from doing that.

4 Counterfactual quantum effects and puzzles

The term *counterfactual* is usually used for things that might have happened, although they did not really happened.

An important point is that while classical counterfactuals do not have physical consequences, quantum counterfactuals can have surprisingly big consequences because the mere possibility that some quantum event might have happened can change the probabilities of obtaining various experimental outcomes.

As a consequence it can be (easily) shown that a quantum computer can provide the result of a computation without performing the computation provided it would provide the same result of computation by really performing the computation (Mitchinson and Jozsa (1999).

Even though the awareness of special effects of quantum counterfactuals was around since 1960, especially in connection with the concept of *interaction-free measurement*, a breakthrough was the paper by Elitzur and Vidman (1993) in which they described a scheme to detect the presence of a bomb which explodes even if a single photon touches its detector. The basic idea is actually extremely simple and uses the well-known Mach-Zehnder interferometer (Figure 4a) with two beam-splitters (half-silver mirrors), and two reflectors (silver mirrors)..

It is straightforward to see that if a single photon enters the interferometer through the indicated input then on the output, because of interference, exactly detector D_1 detects the photon with probability 1 and the photon is never detected by detector D_2.

On the other hand, if a bomb with a supersensitive detector is placed into the lower arm of the interferometer, Figure 4b, in such a way that the incoming photon could hit the bomb detector, it is an easy exercise in superposition to see that with probability $\frac{1}{2}$ the photon hits the bomb detector and the bomb explodes and in this way the presence of the bomb is (in an unfortunate way) detected. However, with probability $\frac{1}{4}$ the photon is detected also by the detector $D1$ and in such a case we only know that the bomb was not touched but we do not know whether the bomb is present. However, with probability $\frac{1}{4}$ the photon is registered in the detector D_2 and in such a case we know not only that bomb was not touched, but also that it is present (in the interferometer scheme9!

The result concerning counterfactual computations can be demonstrated in a similar way by considering in the lower interferometer arm a computer and not a bomb. This can deal with the case that the (quantum) computer has to solve some decision problem, for example to decide whether given

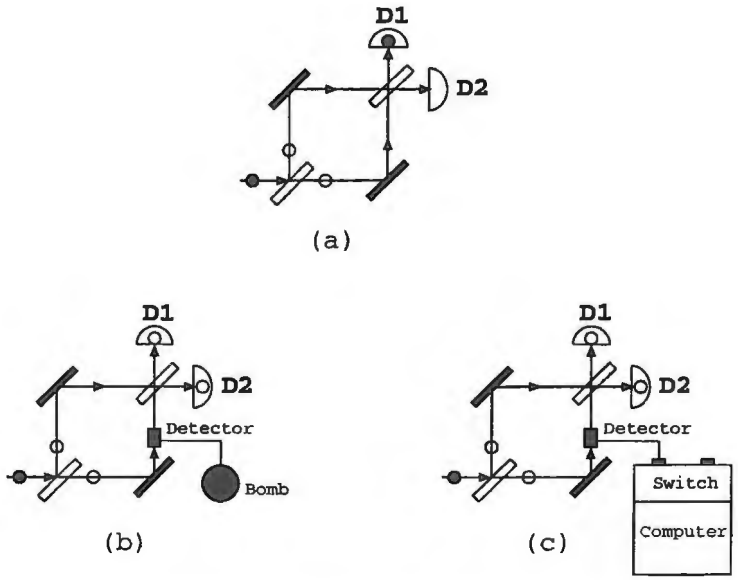


Figure 4: Interaction free measurements

graphs are isomorphic. Computer could have a switch that is either *on* or *off* and a one bit memory register m, with $m = 0$ initially, and the computer changes the contents of the register m to 1 provided it is switched on and it determines that given input graphs are isomorphic. In such a case it can be shown that with non-zero probability we have the case that the computer's switch is off and the computer provides the result of computation ($m = 1$) without doing the computation (if given input graphs are indeed isomorphic).

Closely related to that is the idea of *interaction-free measurements*. They are usually described as measurements in which no energy is exchanged between the measuring device and the measuring object, but the presence of the object is detected anyway. [4] For interaction-free measurements it is essential that the measured object has the potential to absorb the energy from the measuring device, but actually it does not do so.

Attempts have started to explore how to make use of this idea for taking images without touching the objects (White et al. 1998). It is clear that this puzzling idea could have enormous use, provided ...

[4] A caution: this term is actually quite misleading because the fact that no energy is exchanged does not exclude the possibility that some other quantum numbers are exchanged.

5 Quantum effects in the brain?

A somewhat controversial issue is whether quantum effects play any role in the functioning of our brain. Classical neurophysiological models have been succesful in explaining, or at least elucidating, mechanisms in early vision, memory, etc. However, it is felt that there is still a big gap, between what our brain is able to do, and what we know about how it is actually doing it. Though it is too early to reject models based on classical physics, some scientists claim that the full potential of our brains can only be explained by entering the quantum era.

Penrose and Hameroff have proposed the so-called Orch OR (orchestrated objective reduction) model, which posits quantum computation in microtubule protein assemblies in the neurons of the brain (Penrose and Hameroff (1995, 1996), Hameroff (1998)). Microtubules are hollow cylinders, of which the walls consist of the protein tubulin arranged in a skewed hexagonal lattice (Hagan et al. 2000). They are part of the internal scaffolding of cells, and were thought in the past to perform classical information processing, like regulating cellular behavior at the molecular level. At the level of individual proteins, however, quantum computation may play an significant role, as was pointed out in (Conrad, 1994). Microtubules are thought to be important to bridge the gap between tubulin-based quantum computation and the classical functioning of neurons. A possible mechanism for this, described in Voet and Voet (1995), is the formation of structural states of proteins under the influence of quantum van der Waals forces.

In the Penrose-Hameroff model tubulin proteins can maintain coherent superpositions of milliseconds, or even up to a second, before self-collapsing under quantum gravitational influences. This time-frame is relevant in a neurophysiological context, suggesting that cognitive processes can potentially be influenced by quantum coherent states.

This view, however, has been criticized by Tegmark (2000), who pointed out that the quantum superpositions are hard to shield from environmental, and in particular thermal, influences. The brain, being a warm, wet, and noisy organ, is hardly expected to maintain coherence sufficiently long to be under the influence of quantum mechanical mechanisms. This view, on its turn, has been criticized by Hagan et al. (2000).

Whatever the outcome of the debate is, it is worthwhile to consider the following questions:

1. If quantum-superpositions play an important role in the functioning of our brain, how are they affected by measuring brain activity by

means of fMRI, MEG, etc, and how will their possible collapse affect cognitive processes?

It does not seem to have been ever observed that, while being an experimental subject of a measurement, someone ceased to have consciousness, which is often linked to quantum-mechanical effects, or other cognitive functioning was effected. However, the resolution of current measuring devices is so low (at least, not both the spatial and temporal resolutions are high), that it can be imagined that measurements only cause a partial collapse, that is hardly observed by the subject.

2. Will the intra-cellular quantum effects play any role in the interactions between neurons?

   In recent years, more and more neuroscientists started to recognize the computational power that neurons themselves exhibit, as opposed to the traditional view that most of the interesting actions take place in between neurons. The 'Computational Neuron' turns out to be much more than a thresholding device that sums its input, and outputs a binary-like value. Quantum effects may well be able to add significantly to the computational power of neurons, thus also having a big impact on extra-neuronal activity.

3. Are quantum-effects necessary to explain the supposed computational power of the brain beyond Turing machines?

   The view that the neurochemical and electrophysiological mechanisms of synaptic function has a discrete nature, and that cognitive mechanisms merely based on this do not exceed the power of conventional computers, implicating that quantum effects are likely to be significant, can be countered by pointing out the analogue nature of the timing at which spiking of neurons take place. Subtle variations in timing may have a significant meaning, a view which is contrary to the traditional view that the output of neurons should be considered as the time-average of the spikes. Though the analogue characteristic of timing may bring us beyond Turing machines, it should not be excluded that Quantum effects may offer improved efficiency at which calculations can take place, or other advantages.

References

[1] M. Conrad. *Chaos, Solitons and Fractals,* 4: 423– , 1994.

[2] Albert Einstein, Boris Podolsky, and Nathan Rosen. Can quantum mechanical description of physics reality be considered complete? *Physical Review,* 47: 777–780, 1935. Also in *Quantum theory and measurement,* ed. J. A. Wheeler, Wojciech H. Zurek, Princeton University Press, 1983.

[3] A. Elitzur and Lev Vaidman. Quantum-mechanical interaction-free measurements. *Foundations of Physics,* 23: 987–997, 1993.

[4] Jozef Gruska. *Qunatum Computing.* McGraw-Hill, 1999. See also additions and updatings of the book on http://www.mcgraw-hill.co.uk/gruska.

[5] Jozef Gruska. *Mathematics unlimited, 2001 and beyond,* chapter Quantum computing chellenges. Springer, 2000.

[6] S. Hagan, S. R. Hameroff, and J. A. Tuszynski. Quantum computation in brain microtubes? decoherence and biological feasibility. Technical report, quant-ph/0005025, 2000. See also *Physical Review Letters,* 83: 3566– , 1999.

[7] S. R. Hameroff. Quantum Computation in brain microtubes? The Penrose-Hameroff Orch OR model of consciousness. *Transactions of Royal Society (London) A,* 356: 1869– , 1998.

[8] Lucien Hardy. Spooky action at a distance in quantum mechanics. *Contemporary Physics,* 6(6): 419–429, 1998.

[9] I. D. Jonathan and Martin B. Plenio. Entanglement-assisted local manipulation of pure quantum states. Technical report, quant-ph/ ,1999. See also *Physical Review Letters,* 83: 3566– , 1999.

[10] David Lindley. *Where does the weirdness go?* BasicBooks, 1996.

[11] Graeme Mitchison and Richard Jozsa. Counterfactual computation. Technical report, quant-ph/9906026, 1999.

[12] Jian W. Pan, Dik Bouwemester, Harald Weinfurter, and Anton Zeilinger. Experimental entanglement swapping: entangled photons that never interacted. *Physical Review Letters,* 80: 3891–3894, 1998.

[13] Roger Penrose and S. R. Hameroff. What gaps? Reply to Grush and Churchland. *Journal of Consciousness Studies,* 2: 98– , 1995.

[14] Roger Penrose and S. R. Hameroff. Conscious events as orchestrated space-time selections. *Journal of Consciousness Studies,* 3: 356– , 1996.

[15] Ian C. Percival. Why Bell experiments. Technical report, quant-ph/0008097, 2000.

[16] Asher Peres. Delayed choice for entanglement swapping. *Journal of Modern Optics,* 47: 139–143, 2000.

[17] Ran Raz. Exponential seperation of quantum and classical communication complexity. In *Proceedings of 31st ACM STOC,* pp. 358–367, 1999.

[18] John Reif. Alternative computational models: a comparison of biomolecular and quantum computing. In *Proceedings of 18th International Conference on Foundations of Software Technology and Theoretical Computer Science,* pp. 102–121, 1998. For an extended version see http://www.cs.duke/~reif/paper/altcomp.ps.

[19] Andrew M. Steane and Wim van Dam. Guess my number. *Physics Today,* 2000.

[20] M. Tegmark. Importance of quantum decoherence in brain processes. *Physics Review E,* 61: 4194– , 2000.

[21] Wim van Dam, Peter Høyer, and Alain Tapp. Multiparty quantum communication complexity. Technical report, quant-ph/9710054, 1997.

[22] D. Voet and J. G. Voet. *Biochemistry.* Wiley, 1995.

[23] A. White, J. Mitchell, O. Naizr, and P. Kwiat. Interaction-free imaging. Technical report, quant-ph/9803060, 1998.

[24] Marek Žukowski, Anton Zeilinger, Michael A. Horne, and Arhur K. Ekert. "Event-ready-detectors"; Bell experiments via entanglement swapping. *Physical Review Letters,* 71: 4287–4290, 1993.

Quantum Information: the New Frontier

K. Svozil
Institut für Theoretische Physik
University of Technology Vienna
Wiedner Hauptstraße 8-10/136
A-1040 Vienna, Austria
e-mail: svozil@tph.tuwien.ac.at
www: http://tph.tuwien.ac.at/~svozil

http://tph.tuwien.ac.at/~svozil/publ/2000-qitnf.tex .tex

Abstract

Quantum information and computation is the new hype in physics. It is promising, mindboggling and even already applicable in cryptography, with good prospects ahead. A brief, rather subjective outline is presented.

1 Is Nature telling us something?

Friends in experimental physics tell me that the essence of their observations are clicks in some counter. There is a click or there is none. This is experimental physics in a nutshell. There may be some magic in properly designing experiments and some magic in interpreting those clicks, but that is all there is.

A single click represents some elementary physical proposition. It is also an answer to a question which might not even have been posed consciously. It is tempting to state that "Nature wants to tell us something" with these clicks about some formal structures, symmetries, music or numbers beyond the phenomena. Maybe that is the case, and most physicists tend to believe so; but maybe we are just observing crap, erratic emanations devoid of any meaning [1].

Anyway, we have to deal with those clicks, and one way to deal with them is to interpret them as information. For example, in an experimental input-output scheme, information is received, transformed and communicated by

the system. One might think of a physical system as a black box with an input and an output interface [2]. The experimenter inputs some information and the black box responds with some information as output.

If we are dealing with mechanical systems, all the conceivable gadgets inside the black box can be isomorphically translated into a sheet or a tape of paper on which finite computations are performed and vice versa. This was Turing's insight.

But if the black box is essentially quantum driven, then the paper metaphor becomes questionable. The quantum is illusive and highly nonintuitive. In the words of John Archibald Wheeler, one is capturing a "smoky [[quantum]] dragon" [3] inside the black box. Or, in Danny Greenberger's dictum, "quantum mechanics is magic" [4]. In addition, quantized systems such as the quantized electromagnetic field have "more" degrees of freedom as compared to their classical correspondents. Therefore, any isomorphic translation into classical mechanistic devices remains very expensive in terms of paper consumption, at best. To make things worse, under certain reasonable side assumption, it can be proven that a complete "mechanical" paper set of all quantum answers is inconsistent.

Because of these novel non-classical features it is so exiting to pursue the quantum information concept. But even if we look aside and do not want to be bothered with the quantum, the quantum catches up on us: due to the progressing miniaturization of circuits forming logical gates, we shall soon be confronted with quantum phenomena there. In the following, some of the recent developments are reviewed below; and some speculations and prospects are mentioned.

2 Formalization of quantum information

In order to be applicable, any formalization of information has to be based on its proper realization in physical terms; i.e., as states of a physical system. In this view, information theory is part of physics; or conversely, physics is part of information theory. And just as the classical bit represents the distinction between two classical physical states, the quantum bit, henceforth often abbreviated by the term *'qubit,'* represents the conceivable states of the most elementary quantized system. As we shall see, qubits feature quantum mechanics 'in a nutshell.' Quantum bits are more general structures than classical bits. That is, classical bits can be represented as the limit of qubits, but not vice versa.

Classical information theory is based on the classical bit as fundamental

atom. This classical bit, henceforth called *'cbit,'* is in one of two classical states t (often interpreted as "true") and f (often interpreted as "false"). It is customary to code the classical logical states by $\#(t) = 1$ and $\#(f) = 0$ ($\#(s)$ stands for the code of s). The states can, for instance, be realized by some condenser which is discharged ($\equiv$ cbit state 0) or charged ($\equiv$ cbit state 1).

In quantum information theory (see Appendix A for a brief outline of quantum mechanics) qubits can be physically represented by a *'coherent superposition'* of the two orthonormal[1] states t and f. The qubit states

$$x_\alpha = \alpha t + \beta f \tag{1}$$

form a continuum, with $|\alpha|^2 + |\beta|^2 = 1$, $\alpha, \beta \in \mathbf{C}$.

What is a coherent superposition? Formally it is just a sum of two elements (representing quantum states) in Hilbert space, which results in an element (a quantum state) again per definition. So, formally we are on the safe side. Informally speaking, a coherent superposition of two different and classically distinct states contains them both. Now classically this sounds like outright nonsense! A classical bit cannot be true and false at the same time. This would be inconsistent, and inconsistencies in physics sound as absurd as in mathematics [5].

Yet, quantum mechanics (so far consistently) achieves the implementation of classically inconsistent information into a single quantum bit. Why is that possible? Maybe we get a better feeling for this when we take up Erwin Schrödinger's interpretation of the quantum wave function (in our terms: of the qubit states) as a sort of "catalogue of expectation values" [6]. That is, the qubit appears to be a *representation of the state of our knowledge* about a physical system rather than what may be called "its true state." (Indeed we have to be extremely careful here with what we say. The straightforward classical pretension that quantum systems must have "a true state", albeit hidden to us, yields to outright contradictions!)

Why have I mentioned quantum superpositions here? Because they lie at the heart of quantum parallelism. And quantum parallelism lies at the heart of the quantum speedups which caused so much hype recently.

The coding of qubits is discussed in Appendix C.

The classical and the quantum mechanical concept of information differ from each other in several aspects. Intuitively and classically, a unit of information is context-free. That is, it is independent of what other information is or might be present. A classical bit remains unchanged, no matter by

[1] $(t,t) = (f,f) = 1$ and $(t,f) = 0$.

what methods it is inferred. It obeys classical logic. It can be copied. No doubts can be left.

By contrast, to mention just a few nonclassical properties of qubits:

- Qubits are contextual [7]. A quantum bit may appear different, depending on the method by which it is inferred.
- Qubits cannot be copied or "cloned" [8, 9, 10, 11, 12, 13]. This due to the fact that the quantum evolution is reversible, i.e., one-to-one.
- Qubits do not necessarily satisfy classical tautologies such as the distributive law [14, 15].
- Qubits obey quantum logic [16] which is different from classical logic.
- Qubits are coherent superpositions of classically distinct, contradicting information.
- Qubits are subject to complementarity.

3 Complementarity and quantum cryptography

Before we proceed to quantum computing, which makes heavy use of the possibility to superpose classically distinct information, we shall mention an area of quantum information theory which has already matured to the point where the applications have almost become commercially available: quantum cryptography. At the moment, this might be seen as *the* "killer app" of quantum information theory.

Quantum cryptography (for a detailed review see [17]) is based on the quantum mechanical feature of *complementarity.* A formalization of quantum complementarity has been attempted by Edward Moore [18] who started finite automata theory with this. (Recent results are contained in Ref. [19] and [20, chapter 10]; see also Appendix B.)

Informally speaking, quantum complementarity stands for the principal impossibility to measure two observables at the same time with arbitrary position. If you decide to precisely measure the first observable, you "loose control" over the second one and vice versa. By measuring one observable, the state of the system undergoes a "state reduction" or, expressed differently, "the wave function collapses" and becomes different from the original one. This randomizes a subsequent measurement of the second, complementary observable: in performing the subsequent measurement, one obtains some measurement results (i.e., clicks, you remember?), but they dont tell

us much about the original qubit, they are unusable crap. There is no other way of recoving the original state than by completely "undoing" the first measurement in such a way that no trace is left of the previous measurement result; not even a copy of the "classical measurement"[2] result!

So how can this quantum property of complentarity can be put to use in cryptography? The answer is straightforward (if one knows it already): By taking advantage of complementarity, the sender "Alice" of a secret and the receiver "Bob" are able to monitor the secure quantum communication channel and to know when an eavesdropper is present.

This can be done as follows. Assume that Alice sends Bob a qubit and an eavesdropper is present. This eavesdropper is in an inescapable dilemma: neither can the qubit be copied, nor can it be measured. The former case is forbidden in quantum information theory and the letter case would result in a state reduction which modifies Alice's qubit to the point where it is nonsense for Bob. Bob and Alice can realize this by comparing some of their results over a classical (insecure) channel.[3] The exact protocol can for instance be found in [17]. Another scheme [21] operates with entangled pairs of qubits. Here entanglement means that whatever measurement of a particular type is performed on one qubit, if you perform the same measurement on the other qubit of the pair, the result is the same.

Actually, in the real world, the communication over the insecure classical channel has to go back and forth, and they have to constantly compare a certain amount of their measured qubits in order to be able to assure a guaranteed amount of certainty that no eavesdropper is present. That is by no means trivial [22]. But besides this necessary overhead, the quantum channel can be certified to be secure, at least up to some desired amount of certainty and up to the point where someone comes up with a theory which is "better than quantum mechanics" and which circumvents complementarity somehow. Of course, the contemporaries always believe and assure the authorities that there will never be such a theory!

Quantum cryptographic schemes of the above type have already been demonstrated to work for distances of 1000m (and longer) and net key sizes (after error correction) of 59000 Bits at sustained (105 s) production rates of 850 Bits/s [23]. Yet there is no commercially available solution so far.

[2] I put a quote here because if one is able to "undo a measurement", then this process cannot be classical: per definition, classicality means irreversibility, many-to-oneness.

[3] Actually, if the eavesdropper has total control over the classical channel, this might be used for a reasonable attack strategy.

4 Quantum computing

Quantum computers operate with qubits. We have dealt with qubits already. Now what about the operation of quantum computers on qubits? We have to find something similar than Turing's "paper-and-pencil-operations" on paper or tape. The most natural candidate for a formalization is the unitary time evolution of the quantum states. This is all there is (maybe besides measurement [24]), because there is nothing beyond the unitary time evolution. Unitary operators stand for generalized rotations in complex Hilbert spaces. Therefore, a universal quantum computer can just be represented by the most general unitary operator!

That is a straightforward concept: given a finite dimensional Hilbert space of, say, dimension n, then the most general unitary operator $U(n)$ can for instance be parameterized by composition of unitary operations in two (sub)dimensions $U(2)$ [25]. Now we all know how $U(2)$ looks like (cf. Appendix D), so we know how $U(n)$ looks like. Hence we all know how to properly formalize a universal quantum computer!

This looks simple enough, but where is the advantage? Of course one immediate answer is that it is perfectly all right to simulate a quantized system with a quantum computer — we all know that every system is a perfect copy of itself!

But that is not the whole story. What is really challenging here is that we may be able to use quantum parallelism for speedups. And, as mentioned already, at the heart of quantum parallelism is the superposition principle and quantum entanglement. Superposition enables the quantum programmer to "squeeze" 2^N classical bits into N qubits. In processing 1 qubit state $\alpha t + \beta f$, the computer processes 2 classical bit states t and f at once. In processing N qubit states, the computer may be able to processes 2^N classical bit states at once. Many researchers in quantum computing interpret this (in the so-called "Everett interpretation of quantum mechanics") as an indication that 2^N seperate computer run in 2^N seperate worlds (one computer in each world); thereby running through each one of the computational passes in parallel. That might certainly be a big advantage as compared to a classical routine which might only be able to process the cases consecutively, one after the other.

There are indeed indications that speedups are possible. The most prominent examples are Shor's quantum algorithm for prime factoring [26, 27] and Grover's search algorithm [28] for a single item satisfying a given condition in an unsorted database. A detailed review of the suggested quantum algorithms exceeds the scope of this brief discussion and can for

instance be found in Gruska's book [29].

One fundamental feature of the unitary evolution is its bijectivity, its one-to-oneness. This is the reason why copying is not allowed, but this is also the reason why there is no big waste basked where information vanishis into oblivion or nirvana forever. In a quantum computer, one and the same "message" is constantly permutated. It always remains the same but expresses itself through different forms. Information is neither created nor discarded but remains constant at all times.[4]

Is there a price to be pad for parallelism? Let me just mention one important problem here: the problem of the readout of the result. This is no issue in classical computation. But in quantum computation, to use the Everett metaphor, it is by no means trivial how the many parallel entangled universes communicate with each other in such a way that the classical result can be properly communicated. In many cases one has to make sure that, through positive interference, the proper probability amplitudes indicating this result build up. One may even speculate that there is no sufficient buildup of the states if the problem allows for many nonunique solutions [30, 31].

5 Summary and outlook

Let me close with a few observations. So far, quantum information theory has applied the quantum features of complementarity, entanglement and quantum parallelism to more or less real-world applications. Certain other quantum features such as contextuality have not been put to use so far.

There are good prospects for quantum computing; if not for other reasons but because our computer parts will finally reach the quantum domain. We may be just at the very beginning, having conceived the quantum analogies of classical tubes (e.g., quantum optical devices). Maybe in the near future someone comes up with a revolutionary design such as a "quantum transistor" which will radically change the technology of information processing.

This is a very exciting and challenging new field of physics and computer sciences.

[4]This implicit time symmetry spoils the very notion of "progress" or "achievement," since what is a valuable output is purely determined by the subjective meaning the observer associates with it and is devoid of any syntactic relevance.

Appendix A: All (and probably more that) you ever wanted to know about quantum mechanics

"Quantization" has been introduced by Max Planck around 1900 [32, 33, 34]. In a courageous, bold step Planck assumed a *discretization* of the total energy U_N of N linear oscillators ("Resonatoren"),

$$U_N = P\epsilon \in \{0, \epsilon, 2\epsilon, 3\epsilon, 4\epsilon, \ldots\},$$

where $P \in \mathbf{N}_0$ is zero or a positive integer and ϵ stands for the *smallest quantum of energy.* ϵ is a linear function of frequency ω and proportional to Planck's fundamental constant $\hbar \approx 10^{-34}$ Js; i.e.,

$$\epsilon = \hbar\omega.$$

That was a bold step in a time of the predominant continuum models of classical mechanics.

In extension of Planck's discretized resonator energy model, Einstein [35] proposed a quantization of the electromagnetic field. According to the light quantum hypothesis, energy in an electric field mode characterized by the frequency ω can be produced, absorbed and exchanged only in a discrete number n of "lumps" or "quanta" or "photons"

$$E_n = n\hbar\omega\,,\ n = 0, 1, 2, 3, \ldots\ .$$

The following is a very brief introduction to the principles of quantum mechanics for logicians and computer scientists, as well as a reminder for physicists.[5] To avoid a shock from a too early exposure to "exotic" nomenclature prevalent in physics — the Dirac bra-ket notation — the notation of Dunford-Schwartz [49] is adopted.[6]

Quantum mechanics, just as classical mechanics, can be formalized in terms of a linear space structure, in particular by Hilbert spaces [45]. That is, all objects of quantum physics, in particular the ones used by quantum logic,

[5]Introductions to quantum mechanics can be found in Feynman, Leighton & M. Sands [36], Harris [37], Lipkin [38], Ballentine [39], Messiah [40], Davydov [41], Dirac [42], Peres [43], Mackey [44], von Neumann [45], and Bell [46], among many other expositions. The history of quantum mechanics is reviewed by Jammer [47]. Wheeler & Zurek [48] published a helpful resource book.

[6]The bra-ket notation introduced by Dirac is widely used in physics. To translate expressions into the bra-ket notation, the following identifications work for most practical purposes: for the scalar product, "$(\equiv$ (", "$\rangle \equiv)$", "$, \equiv |$". States are written as $|\,\psi\rangle \equiv \psi$, operators as $\langle i \mid A \mid j\rangle \equiv A_{ij}$.

ought to be expressed in terms of objects based on concepts of Hilbert space theory—scalar products, linear summations, subspaces, operators, measures and so on.

Unless stated differently, only finite-dimensional Hilbert spaces are considered.[7]

A quantum mechanical *Hilbert space* is a linear vector space $\mathcal{H}$ over the field $\mathbf{C}$ of complex numbers (with vector addition and scalar multiplication), together with a complex function $(\cdot,\cdot)$, the *scalar* or *inner product*, defined on $\mathcal{H} \times \mathcal{H}$ such that (i) $(x,x) = 0$ if and only if $x = 0$; (ii) $(x,x) \geq 0$ for all $x \in \mathcal{H}$; (iii) $(x+y,z) = (x,z) + (y,z)$ for all $x,y,z \in \mathcal{H}$; (iv) $(\alpha x, y) = \alpha(x,y)$ for all $x,y \in \mathcal{H}, \alpha \in \mathbf{C}$; (v) $(x,y) = (y,x)^*$ for all $x,y \in \mathcal{H}$ (α^* stands for the complex conjugate of α); (vi) If $x_n \in \mathcal{H}$, $n = 1,2,\ldots$, and if $\lim_{n,m\to\infty}(x_n - x_m, x_n - x_m) = 0$, then there exists an $x \in \mathcal{H}$ with $\lim_{n\to\infty}(x_n - x, x_n - x) = 0$.

We shall make the following identifications between physical and theoretical objects (a *caveat:* this is an incomplete list).

(0) The dimension of the Hilbert space corresponds to the number of degrees of freedom.

(I) A *pure physical state* x is represented either by the one-dimensional linear subspace (closed linear manifold) $(x) = \{y \mid y = \alpha x,\ \alpha \in \mathbf{C},\ x \in \mathcal{H}\}$ spanned by a (normalized) vector x of the Hilbert space $\mathcal{H}$ or by the orthogonal projection operator E_x onto (x). Thus, a vector $x \in \mathcal{H}$ represents a pure physical state.

Every one-dimensional projection E_x onto a one-dimensional linear subspace (x) spanned by $x \in \mathcal{H}$ can be represented by the dyadic product $E_x = |x)(x|$.

If two nonparallel vectors $x,y \in \mathcal{H}$ represent pure physical states, their vector sum $z = x+y \in \mathcal{H}$ is again a vector representing a pure physical state. This state z is called the *superposition* of state x and y.[8]

[7] Infinite dimensional cases and continuous spectra are nontrivial extensions of the finite dimensional Hilbert space treatment. As a heuristic rule, which is not always correct, it might be stated that the sums become integrals, and the Kronecker delta function δ_{ij} becomes the Dirac delta function $\delta(i-j)$, which is a generalized function in the continuous variables i,j. In the Dirac bra-ket notation, unity is given by $\mathbf{1} = \int_{-\infty}^{+\infty} |i)(i|\, di$. For a careful treatment, see, for instance, the books by Reed and Simon [50, 51].

[8] $x + y$ is sometimes referred to as "coherent" superposition to indicate the difference to "incoherent" mixtures of state vectors, in which the absolute squares $|x|^2 + |y|^2$ are summed up.

Elements $b_i, b_j \in \mathcal{H}$ of the set of orthonormal base vectors satisfy $(b_i, b_j) = \delta_{ij}$, where $\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$ is the Kronecker delta function. Any pure state x can be written as a linear combination of the set of orthonormal base vectors $\{b_1, b_2, \cdots\}$, i.e., $x = \sum_{i=1}^{n} \beta_i b_i$, where n is the dimension of $\mathcal{H}$ and $\beta_i = (b_i, x) \in \mathbf{C}$. In the Dirac bra-ket notation, unity is given by $\mathbf{1} = \sum_{i=1}^{n} |b_i)(b_i|$.

In the nonpure state case, the system is characterized by the density operator ρ, which is nonnegative and of trace class.[9] If the system is in a nonpure state, then the preparation procedure does not specify the decomposition into projection operators (depending on the choice of basis) precisely. ρ can be brought into its spectral form $\rho = \sum_{i=1}^{n} P_i E_i$, where E_i are projection operators and the P_i's are the associated probabilities (nondegenerate case[10]).

(II) *Observables* A are represented by hermitian operators A on the Hilbert space $\mathcal{H}$ such that $(Ax, y) = (x, Ay)$ for all $x, y \in \mathcal{H}$. (Observables and their corresponding operators are identified.) In matrix notation, the adjoint matrix $A^\dagger$ is the complex conjugate of the transposed matrix of A; i.e., $(A^\dagger)_{ij} = (A^*)_{ji}$. Hermiticity means that $(A^\dagger)_{ij} = A_{ij}$.

Any hermitian operator has a spectral representation $A = \sum_{i=1}^{n} \alpha_i E_i$, where the E_i's are orthogonal projection operators onto the orthonormal eigenvectors a_i of A (nondegenerate case).

Note that the projection operators, as well as their corresponding vectors and subspaces, have a double rôle as pure state and elementary proposition (that the system is in that pure state).

Observables are said to be *compatible* or *comeasurable* if they can be defined simultaneously with arbitrary accuracy. Compatible observables are polynomials (Borel measurable functions in the infinite dimensional case) of a single "Ur"-observable.

A criterion for compatibility is the *commutator*. Two observables A, B are compatible if their *commutator* vanishes; i.e., if $[A, B] = AB - BA = 0$. In this case, the hermitian matrices A and B can be simultaneously diagonalized, symbolizing that the observables corre-

[9] Nonnegativity means $(\rho x, x) = (x, \rho x) \geq 0$ for all $x \in \mathcal{H}$, and trace class means $\text{trace}(\rho) = 1$.

[10] If the same eigenvalue of an operator occurs more than once, it is called *degenerate*.

sponding to A and B are simultaneously measurable.[11]

It has recently been demonstrated that (by an analog embodiment using particle beams) every hermitian operator in a finite dimensional Hilbert space can be experimentally realized [52].

Actually, one can also measure normal operators N which can be decomposed into the sum of two commuting operators A, B according to $N = A + iB$, with $[A, B] = 0$.

(III) The result of any single measurement of the observable A on an arbitrary state $x \in \mathcal{H}$ can only be one of the real eigenvalues of the corresponding hermitian operator A. (Actually, one can also measure normal operators which can be decomposed into the sum of two commuting If $x = \beta_1 a_1 + \cdots + \beta_i a_i + \cdots + \beta_n a_n$ is in a superposition of eigenstates $\{a_1, \ldots, a_n\}$ of A, the particular outcome of any such single measurement is indeterministic; i.e., it cannot be predicted with certainty. As a result of the measurement, the system is in the state a_i which corresponds to the associated real-valued eigenvalue α_i which is the measurement outcome; i.e.,

$$x \rightarrow a_i.$$

The arrow symbol "$\rightarrow$" denotes an irreversible measurement; usually interpreted as a "transition" or "reduction" of the state due to an irreversible interaction of the microphysical quantum system with a classical, macroscopic measurement apparatus. This "reduction" has given rise to speculations concerning the "collapse of the wave function (state)."

As has been argued recently (e.g., by Greenberger and YaSin [53], and by Herzog, Kwiat, Weinfurter and Zeilinger [54]), it is possible to reconstruct the state of the physical system before the measurement; i.e., to "reverse the collapse of the wave function," if the process of measurement is reversible. After this reconstruction, no information about the measurement is left, not even in principle.

---

[11] Let us first diagonalize A; i.e., $A_{ij} = \text{diag}\,(A_{11}, A_{22}, \ldots, A_{nn})_{ij} = \begin{cases} A_{ii} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$. Then, if A commutes with B, the commutator $[A, B]_{ij} = (AB - BA)_{ij} = A_{ik}B_{ki} - B_{ik}A_{kj} = (A_{ii} - A_{jj})B_{ij} = 0$ vanishes. If A is nondegenerate, then $A_{ii} \neq A_{jj}$ and thus $B_{ij} = 0$ for $i \neq j$. In the degenerate case, B can only be block diagonal. That is, each one of the blocks of B corresponds to a set of equal eigenvalues of A such that the corresponding subblockmatrix of A is proportional to the unit matrix. Thus, each block of B can be diagonalized separately without affecting A [43, p. 71].

How did Schrödinger, the creator of wave mechanics, perceive the quantum physical state, or, more specifically, the ψ-function? In his 1935 paper "Die gegenwärtige Situation in der Quantenmechanik" ("The present situation in quantum mechanics" [6, p. 823]), Schrödinger states,[12]

> *The ψ-function as expectation-catalog:* ... In it [[the ψ-function]] is embodied the momentarily-attained sum of theoretically based future expectation, somewhat as laid down in a *catalog.* ... For each measurement one is required to ascribe to the ψ-function (=the prediction catalog) a characteristic, quite sudden change, which *depends on the measurement result obtained,* and so *cannot be foreseen;* from which alone it is already quite clear that this second kind of change of the ψ-function has nothing whatever in common with its orderly development *between* two measurements. The abrupt change [[of the ψ-function (=the prediction catalog)]] by measurement ... is the most interesting point of the entire theory. It is precisely *the* point that demands the break with naive realism. For *this* reason one cannot put the ψ-function directly in place of the model or of the physical thing. And indeed not because one might never dare impute abrupt unforeseen changes to a physical thing or to a model, but because in the realism point of view observation is a natural process like any other and cannot *per se* bring about an interruption of the orderly flow of natural events.

It therefore seems not unreasonable to state that, epistemologically,

[12] *Die ψ-Funktion als Katalog der Erwartung:* ... Sie [[die ψ-Funktion]] ist jetzt das Instrument zur Voraussage der Wahrscheinlichkeit von Maßzahlen. In ihr ist die jeweils erreichte Summe theoretisch begründeter Zukunftserwartung verkörpert, gleichsam wie in einem *Katalog* niedergelegt. ... Bei jeder Messung ist man genötigt, der ψ-Funktion (=dem Voraussagenkatalog) eine eigenartige, etwas plötzliche Veränderung zuzuschreiben, die von der *gefundenen Maßzahl* abhängt und sich *nicht vorhersehen läßt;* woraus allein schon deutlich ist, daß diese zweite Art von Veränderung der ψ-Funktion mit ihrem regelmäßigen Abrollen *zwischen* zwei Messungen nicht das mindeste zu tun hat. Die abrupte Veränderung durch die Messung ... ist der interessanteste Punkt der ganzen Theorie. Es ist genau *der* Punkt, der den Bruch mit dem naiven Realismus verlangt. Aus *diesem* Grund kann man die ψ-Funktion *nicht* direkt an die Stelle des Modells oder des Realdings setzen. Und zwar nicht etwa weil man einem Realding oder einem Modell nicht abrupte unvorhergesehene Änderungen zumuten dürfte, sondern weil vom realistischen Standpunkt die Beobachtung ein Naturvorgang ist wie jeder andere und nicht per se eine Unterbrechung des regelmäßigen Naturlaufs hervorrufen darf.

quantum mechanics appears more as a theory of knowledge of an (intrinsic) observer rather than the Platonic physics "God knows." The wave function, i.e., the state of the physical system in a particular representation (base), is a representation of the observer's knowledge; it is a representation or name or code or index of the information or knowledge the observer has access to.

(IV) The probability $P_x(y)$ to find a system represented by a normalized pure state x in some normalized pure state y is given by

$$P_x(y) = |(x, y)|^2, \quad |x|^2 = |y|^2 = 1.$$

In the nonpure state case, The probability $P(y)$ to find a system characterized by ρ in a pure state associated with a projection operator E_y is

$$P_\rho(y) = \text{trace}(\rho E_y).$$

(V) The *average value* or *expectation value* of an observable A represented by a hermitian operator A in the normalized pure state x is given by

$$\langle A \rangle_x = \sum_{i=1}^{n} \alpha_i |(x, a_i)|^2, \quad |x|^2 = |a_i|^2 = 1.$$

The *average value* or *expectation value* of an observable A represented by a hermitian operator A in the nonpure state ρ is given by

$$\langle A \rangle = \text{trace}(\rho A) = \sum_{i=1}^{n} \alpha_i \text{trace}(\rho E_i).$$

(VI) The dynamical law or equation of motion between subsequent, irreversible, measurements can be written in the form $x(t) = Ux(t_0)$, where $U^\dagger = U^{-1}$ ("$\dagger$ stands for transposition and complex conjugation) is a linear *unitary evolution operator.*[13] Per definition, this evolution is reversible; i.e., bijective, one-to-one. So, in quantum mechanics we have to distinguish between unitary, reversible evolution of the system inbetween measurements, and the "collapse of the wave function" at an irreversible measurement.

[13] Any unitary operator $U(n)$ in finite-dimensional Hilbert space can be represented by the product — the serial composition — of unitary operators $U(2)$ acting in twodimensional subspaces [25, 52].

The *Schrödinger equation* $i\hbar\frac{\partial}{\partial t}\psi(t) = H\psi(t)$ for some state ψ is obtained by identifying U with $U = e^{-iHt/\hbar}$, where H is a hermitian Hamiltonian ("energy") operator, by partially differentiating the equation of motion with respect to the time variable t; i.e., $\frac{\partial}{\partial t}\psi(t) = -\frac{iH}{\hbar}e^{-iHt/\hbar}\psi(t_0) = -\frac{iH}{\hbar}\psi(t)$. In terms of the set of orthonormal base vectors $\{b_1, b_2, \ldots\}$, the Schrödinger equation can be written as $i\hbar\frac{\partial}{\partial t}(b_i, \psi(t)) = \sum_j H_{ij}(b_j, \psi(t))$.

For stationary states $\psi_n(t) = e^{-(i/\hbar)E_n t}\psi_n$, the Schrödinger equation can be brought into its time-independent form $H\,\psi_n = E_m\,\psi_m$ (nondegenerate case). Here, $i\hbar\frac{\partial}{\partial t}\psi_m(t) = E_m\,\psi_m(t)$ has been used; E_m and ψ_m stand for the m'th eigenvalue and eigenstate of H, respectively.

Usually, a physical problem is defined by the Hamiltonian H and the Hilbert space in question. The problem of finding the physically relevant states reduces to finding a complete set of eigenvalues and eigenstates of H.

Appendix B: Complementarity and automaton logic

A systematic, formal investigation of the black box system or any finite input/output system can be given by finite automata. Indeed, the study of finite automata was motivated from the very beginning by their analogy to quantum systems [18]. Finite automata are universal with respect to the class of computable functions. That is, universal networks of automata can compute any effectively (Turing-) computable function. Conversely, any feature emerging from finite automata is reflected by any other universal computational device. In this sense, they are "robust". All rationally conceivable finite games can be modeled by finite automata.

Computational complementarity, as it is sometimes called [55], can be introduced as a game between Alice and Bob. The rules of the game are as follows. Before the actual game, Alice gives Bob all he needs to know about the intrinsic workings of the automaton. For example, Alice tells Bob, *"if the automaton is in state 1 and you input the symbol 2, then the automaton will make a transition into state 2 and output the symbol 0,"* and so on. Then Alice presents Bob a black box which contains a realization of the automaton. Attached to the black box are two interfaces: a keyboard for the input of symbols, and an output display, on which the output symbols appear. Again, no other interfaces are allowed. In particular, Bob is not allowed to "screw the box open."

Suppose now that Alice chooses some initial state of the automaton. She may either throw a dice, or she may make clever choices using some formalized system. In any case, Alice does not tell Bob about her choice. All Bob has at his disposal are the input-output interfaces.

Bob's goal is to find out which state Alice has chosen. Alice's goal is to fool Bob.

Bob may simply guess or rely on his luck by throwing a dice. But Bob can also perform clever input-output experiments and analyze his data in order to find out. Bob wins if he gives the correct answer. Alice wins if Bob's guess is incorrect. (So, Alice has to be really mean and select worst-case scenarios).

Suppose that Bob tries very hard. Is cleverness sufficient? Will Bob always be able to uniquely determine the initial automaton state?

The answer to that question is "no." The reason is that there may be situations when Bob's input causes an irreversible transition into a black box state which does not allow any further queries about the initial state.

What has been introduced here as a game between Alice and Bob is what the mathematicians have called the *state identification problem* [18, 56, 57, 58]: given a finite deterministic automaton, the task is to locate an unknown initial state. Thereby it is assumed that only *a single* automaton copy is available for inspection. That is, no second, identical, example of the automaton can be used for further examination. Alternatively, one may think of it as choosing at random a single automaton from a collection of automata in an ensemble differing only by their initial state. The task then is to find out which was the initial state of the chosen automaton.

The logico-algebraic structure of the state identification problem has been introduced in [59], and subsequently studied in [59, 60, 61, 62, 63, 64, 65, 19]. We shall deal with it next.

Step 1: Computation of the experimental equivalence classes.

In the propositional structure of sequential machines, state partitions play an important rôle. Indeed, the set of states is partitioned into equivalence classes with respect to a particular input-output experiment.

Suppose again that the only unknown feature of an automaton is its initial state; all else is known. The automaton is presented in a black box, with input and output interfaces. The task in this *complementary game* is to find (partial) information about the initial state of the automaton [18].

To illustrate this, consider the Mealy automaton M_s discussed above. Input/output experiments can be performed by the input of just one symbol

i (in this example, more inputs yield no finer partitions). Suppose again that Bob does not know the automaton's initial state. So, Bob has to choose between the input of symbols 1,2, or 3. If Bob inputs, say, symbol 1, then he obtains a definite answer whether the automaton was in state 1 — corresponding to output 1; or whether the automaton was not in state 1 — corresponding to output 0. The latter proposition "not 1" can be identified with the proposition that the automaton was either in state 2 or in state 3.

Likewise, if Bob inputs symbol 2, he obtains a definite answer whether the automaton was in state 2 — corresponding to output 1; or whether the automaton was not in state 2 — corresponding to output 0. The latter proposition "not 2" can be identified with the proposition that the automaton was either in state 1 or in state 3. Finally, if Bob inputs symbol 3, he obtains a definite answer whether the automaton was in state 3 — corresponding to output 1; or whether the automaton was not in state 3 — corresponding to output 0. The latter proposition "not 3" can be identified with the proposition that the automaton was either in state 1 or in state 2.

Recall that Bob can actually perform only one of these input-output experiments. This experiment will irreversibly destroy the initial automaton state (with the exception of a "hit"; i.e., of output 1). Let us thus describe the three possible types of experiment as follows.

- Bob inputs the symbol 1.
- Bob inputs the symbol 2.
- Bob inputs the symbol 3.

The corresponding observable propositions are:

$p_{\{1\}} \equiv \{1\}$: On input 1, Bob receives the output symbol 1.

$p_{\{2,3\}} \equiv \{2,3\}$: On input 1, Bob receives the output symbol 0.

$p_{\{2\}} \equiv \{2\}$: On input 2, Bob receives the output symbol 1.

$p_{\{1,3\}} \equiv \{1,3\}$: On input 2, Bob receives the output symbol 0.

$p_{\{3\}} \equiv \{3\}$: On input 3, Bob receives the output symbol 1.

$p_{\{1,2\}} \equiv \{1,2\}$: On input 3, Bob receives the output symbol 0.

Note that, in particular, $p_{\{1\}}, p_{\{2\}}, p_{\{3\}}$ are not comeasurable. Note also that, for $\epsilon_{ijk} \neq 0$, $p'_{\{i\}} = p_{\{j,k\}}$ and $p_{\{j,k\}} = p'_{\{i\}}$; or equivalently $\{i\}' = \{j,k\}$ and $\{j,k\} = \{i\}'$.

In that way, we naturally arrive at the notion of a *partitioning* of automaton states according to the information obtained from input/output experiments. Every element of the partition stands for the proposition that the automaton is in (one of) the state(s) contained in that partition. Every partition corresponds to a quasi-classical Boolean block. Let us denote by $v(x)$ the block corresponding to input (sequence) x. Then we obtain

no input:

$$v(\emptyset) = \{\{1,2,3\}\},$$

one input symbol:

input		output 1		output 0
$v(1)$	$=$	$\{\{1\}$	,	$\{2,3\}\}$
$v(2)$	$=$	$\{\{2\}$	,	$\{1,3\}\}$
$v(3)$	$=$	$\{\{3\}$	,	$\{1,2\}\}$.

Conventionally, only the finest partitions are included into the set of state partitions.

Step 2: Pasting of the partitions.

Just as in quantum logic, the *automaton propositional calculus* and the associated *partition logic* is the *pasting* of all the blocks of partitions $v(i)$ on the atomic level. That is, elements of two blocks are identified if and only if the corresponding atoms are identical.

The automaton partition logic based on *atomic* pastings differs from previous approaches [59, 60, 61, 62, 63, 64, 65, 19]. Atomic pasting guarantees that there is no mixing of elements belonging to two different order levels. Such confusions can give rise to the nontransitivity of the order relation [59] in cases where both $p \to q$ and $q \to r$ are operational but incompatible, i.e., complementary, and hence $p \to r$ is not operational.

For the Mealy automaton M_s discussed above, the pasting renders just the horizontal sum — only the least and greatest elements $0, 1$ of each 2^2 is identified—and one obtains a "Chinese lantern" lattice MO_3. The Hasse diagram of the propositional calculus is drawn in Figure 1.

Let us give a formal definition for the procedures sketched so far. Assume a set S and a family of partitions $\mathcal{B}$ of S. Every partition $E \in \mathcal{B}$ can be identified with a Boolean algebra B_E in a natural way by identifying the elements of the partition with the atoms of the Boolean algebra. The pasting

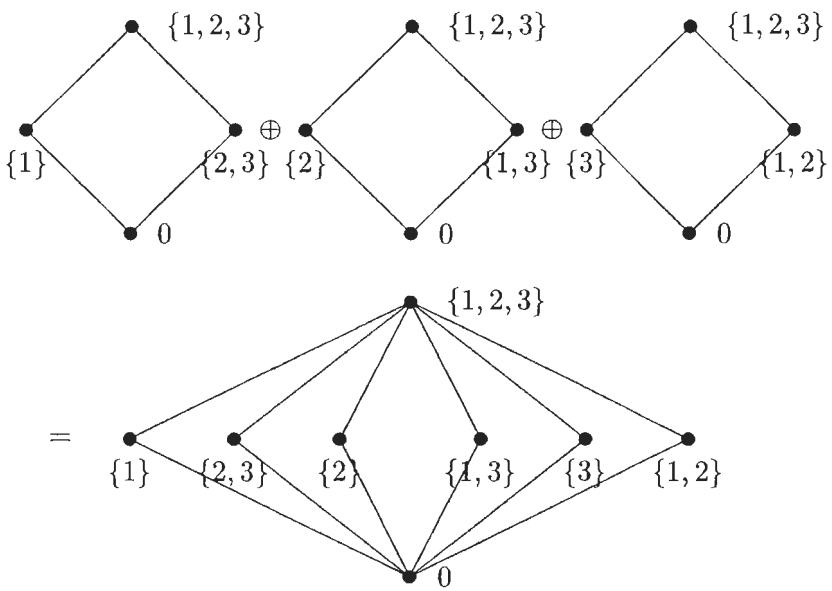


Figure 1: Hasse diagram of the propositional calculus of the Mealy automaton.

of the Boolean algebras $B_E, E \in \mathcal{B}$ on the atomic level is called a partition logic, denoted by $(S, \mathcal{B})$.

The logical structure of the complementarity game (initial-state identification problem) can be defined as follows. Let us call a proposition concerning the initial state of the machine *experimentally decidable* if there is an experiment E which determines the truth value of that proposition. This can be done by performing E, i.e., by the input of a sequence of input symbols $i_1, i_2, i_3, \ldots, i_n$ associated with E, and by observing the output sequence

$$\lambda_E(s) = \lambda(s, i_1), \lambda(\delta(s, i_1), i_2), \ldots, \lambda(\underbrace{\delta(\cdots \delta(s, i_1) \cdots, i_{n-1})}_{n-1 \text{ times}}, i_n).$$

The most general form of a prediction concerning the initial state s of the machine is that the initial state s is contained in a subset P of the state set S. Therefore, we may identify propositions concerning the initial state with subsets of S. A subset P of S is then identified with the proposition that the initial state is contained in P.

Let E be an experiment (a preset or adaptive one), and let $\lambda_E(s)$ denote

the obtained output of an initial state s. λ_E defines a mapping of S to the set of output sequences O^*. We define an equivalence relation on the state set S by

$$s \overset{E}{\equiv} t \text{ if and only if } \lambda_E(s) = \lambda_E(t)$$

for any $s, t \in S$. We denote the partition of S corresponding to $\overset{E}{\equiv}$ by $S/\overset{E}{\equiv}$. Obviously, the propositions decidable by the experiment E are the elements of the Boolean algebra generated by $S/\overset{E}{\equiv}$, denoted by B_E.

There is also another way to construct the experimentally decidable propositions of an experiment E. Let $\lambda_E(P) = \bigcup_{s \in P} \lambda_E(s)$ be the direct image of P under λ_E for any $P \subseteq S$. We denote the direct image of S by O_E; i.e., $O_E = \lambda_E(S)$.

It follows that the most general form of a prediction concerning the outcome W of the experiment E is that W lies in a subset of O_E. Therefore, the experimentally decidable propositions consist of all inverse images $\lambda_E^{-1}(Q)$ of subsets Q of O_E, a procedure which can be constructively formulated (e.g., as an effectively computable algorithm), and which also leads to the Boolean algebra B_E.

Let $\mathcal{B}$ be the set of all Boolean algebras B_E. We call the partition logic $R = (S, \mathcal{B})$ an *automaton propositional calculus.*

Appendix C: Quantum coding

In the usual Hilbert space formulization, qubits can then be written as

$$\#(x_\alpha) = e^{i\varphi}(\sin\omega, e^{i\delta}\cos\omega) \in \mathbf{C}^2, \tag{2}$$

with $\alpha = \alpha(\omega, \varphi, \delta)$, $\omega, \varphi, \delta \in \mathbf{R}$ Qubits can be identified with cbits as follows

$$\#(x_{\alpha(\pi/2,\varphi,\delta)}) = (a, 0) \equiv 1 \text{ and } \#(x_{\alpha(0,\varphi,\delta)}) = (0, b) \equiv 0 \quad , \qquad |a|, |b| = 1 \quad , \tag{3}$$

where the complex numbers a and b are of modulus one. The quantum mechanical states associated with the classical states 0 and 1 are mutually orthogonal.

Notice that, provided that $\alpha, \beta \neq 0$, a qubit is not in a pure classical state. Therefore, any practical determination of the qubit x_α amounts to a measurement of the state amplitude of t or f. According to the quantum postulates, any such *single* measurement will be indeterministic (provided again that $\alpha, \beta \neq 0$). That is, the outcome of a single measurement occurs unpredictably. The probabilities that the qubit x_α is measured in states

t and f are $P_t(x_\alpha) = |(x_\alpha, t)|^2$ and $P_f(x_\alpha) = |(x_\alpha, f)|^2 = 1 - P_t(_{\alpha,\beta})$, respectively.

Appendix D: Universal manipulation of a single qubit: the $U(2)$-gate

It is well known that any n-dimensional unitary matrix U can be composed from elementary unitary transformations in two-dimensional subspaces of $\mathbf{C}^n$. This is usually shown in the context of parameterization of the n-dimensional unitary groups (cf. [25, chapter 2] and [52, 66]). Thereby, a transformation in n-dimensional spaces is decomposed into transformations in 2-dimensional subspaces. This amounts to a successive array of $U(2)$ elements, which in their entirety forms an arbitrary time evolution $U(n)$ in n-dimensional Hilbert space.

Hence, all quantum processes and computation tasks which can possibly be executed must be representable by unitary transformations. Indeed, unitary transformations of qubits are a necessary and sufficient condition for quantum computing. *The group of unitary transformations in arbitrary- but finite-dimensional Hilbert space is a model of universal quantum computer.*

It remains to be shown that the universal $U(2)$-gate is physically operationalizable. This can be done in the framework of Mach-Zehnder interferometry. Note that the number of elementary $U(2)$-transformations is polynomially bounded and does not exceed $\begin{pmatrix} n \\ 2 \end{pmatrix} = n\,(n-1)/2 = O(n^2)$.

In what follows, a lossless *Mach-Zehnder* interferometer drawn in Fig. 2 is discussed. The computation proceeds by successive substitution (transition) of states; i.e.,

$$S_1 : a \rightarrow (b + ic)/\sqrt{2} \quad , \tag{4}$$

$$P : b \rightarrow be^{i\varphi} \quad , \tag{5}$$

$$S_2 : b \rightarrow (c + id)/\sqrt{2} \quad , \tag{6}$$

$$S_2 : c \rightarrow (d + ie)/\sqrt{2} \quad . \tag{7}$$

The resulting transition is

$$a \rightarrow \psi = i \left(\frac{e^{i\varphi} + 1}{2} \right) d + \left(\frac{e^{i\varphi} - 1}{2} \right) e \quad . \tag{8}$$

Assume that $\varphi = 0$, i.e., there is no phase shift at all. Then, equation (8) reduces to $a \rightarrow id$, and the emitted quant is detected only by D_1. Assume

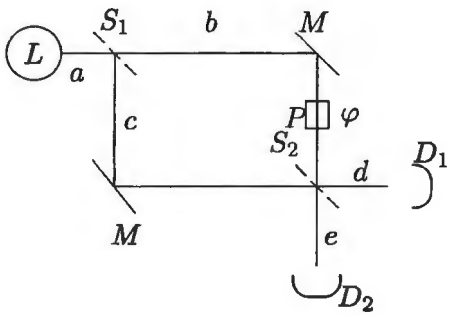


Figure 2: Mach-Zehnder interferometer. A single quantum (photon, neutron, electron *etc*) is emitted in L and meets a lossless beam splitter (half-silvered mirror) S_1, after which its wave function is in a coherent superposition of b and c. In beam path b a phase shifter shifts the phase of state b by φ. The two beams are then recombined at a second lossless beam splitter (half-silvered mirror) S_2. The quant is detected at either D_1 or D_2, corresponding to the states d and e, respectively.

that $\varphi = \pi$. Then, equation (8) reduces to $a \rightarrow -e$, and the emitted quant is detected only by D_2. If one varies the phase shift φ, one obtains the following detection probabilities:

$$P_{D_1}(\varphi) = |(d, \psi)|^2 = \cos^2(\frac{\varphi}{2}) \quad , \quad P_{D_2}(\varphi) = |(e, \psi)|^2 = \sin^2(\frac{\varphi}{2}) \quad . \tag{9}$$

For some "mindboggling" features of Mach-Zehnder interferometry, see [67].

The elementary quantum interference device $\mathbf{T}_{21}^{bs}$ depicted in Fig. (3.a) is just a beam splitter followed by a phase shifter in one of the output ports.

Alternatively, the action of a lossless beam splitter may be described by the matrix $\begin{pmatrix} T(\omega) & i\,R(\omega) \\ i\,R(\omega) & T(\omega) \end{pmatrix} = \begin{pmatrix} \cos\omega & i\,\sin\omega \\ i\,\sin\omega & \cos\omega \end{pmatrix}$. A phase shifter in a two-dimensional Hilbert space is represented by either $\begin{pmatrix} e^{i\varphi} & 0 \\ 0 & 1 \end{pmatrix}$ or $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix}$. The action of the entire device consisting of such elements is calculated by multiplying the matrices in reverse order in which the quanta pass these elements [68, 69].

$$P_1 : \mathbf{0} \rightarrow \mathbf{0}e^{i\alpha+\beta} \quad , \tag{10}$$

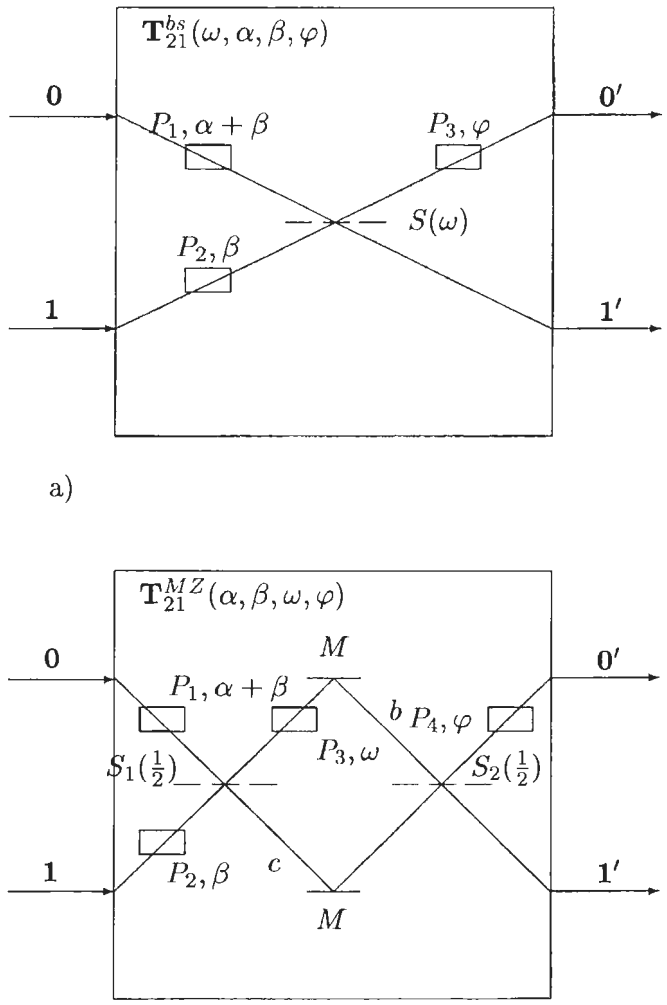


Figure 3: Elementary quantum interference device. An elementary quantum interference device can be realized by a 4-port interferometer with two input ports $\mathbf{0}, \mathbf{1}$ and two output ports $\mathbf{0}', \mathbf{1}'$. Any two-dimensional unitary transformation can be realized by the devices. a) shows a realization by a single beam splitter $S(T)$ with variable transmission t and three phase shifters P_1, P_2, P_3; b) shows a realization with 50:50 beam splitters $S_1(\frac{1}{2})$ and $S_2(\frac{1}{2})$ and four phase shifters P_1, P_2, P_3, P_4.

$$P_2 : \mathbf{1} \rightarrow \mathbf{1}e^{i\beta} \quad , \tag{11}$$
$$S : \mathbf{0} \rightarrow T\,\mathbf{1}' + iR\,\mathbf{0}' \quad , \tag{12}$$
$$S : \mathbf{1} \rightarrow T\,\mathbf{0}' + iR\,\mathbf{1}' \quad , \tag{13}$$
$$P_3 : \mathbf{0}' \rightarrow \mathbf{0}'e^{i\varphi} \quad . \tag{14}$$

If $\mathbf{0} \equiv \mathbf{0}' \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\mathbf{1} \equiv \mathbf{1}' \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $R(\omega) = \sin\omega$, $T(\omega) = \cos\omega$, then the corresponding unitary evolution matrix which transforms any coherent superposition of $\mathbf{0}$ and $\mathbf{1}$ into a superposition of $\mathbf{0}'$ and $\mathbf{1}'$ is given by

$$\begin{aligned} \mathbf{T}_{21}^{bs}(\omega,\alpha,\beta,\varphi) &= \left[e^{i\,\beta} \begin{pmatrix} i\,e^{i(\alpha+\varphi)}\sin\omega & e^{i\alpha}\cos\omega \\ e^{i\varphi}\cos\omega & i\sin\omega \end{pmatrix} \right]^{-1} \\ &= e^{-i\,\beta} \begin{pmatrix} -i\,e^{-i(\alpha+\varphi)}\sin\omega & e^{-i\varphi}\cos\omega \\ e^{-i\alpha}\cos\omega & -i\sin\omega \end{pmatrix} \quad . \end{aligned} \tag{15}$$

The elementary quantum interference device $\mathbf{T}_{21}^{MZ}$ depicted in Fig. (3.b) is a (rotated) Mach-Zehnder interferometer with *two* input and output ports and three phase shifters. According to the "toolbox" rules, the process can be quantum mechanically described by

$$P_1 : \mathbf{0} \rightarrow \mathbf{0}e^{i\alpha+\beta} \quad , \tag{16}$$
$$P_2 : \mathbf{1} \rightarrow \mathbf{1}e^{i\beta} \quad , \tag{17}$$
$$S_1 : \mathbf{1} \rightarrow (b + i\,c)/\sqrt{2} \quad , \tag{18}$$
$$S_1 : \mathbf{0} \rightarrow (c + i\,b)/\sqrt{2} \quad , \tag{19}$$
$$P_3 : c \rightarrow ce^{i\omega} \quad , \tag{20}$$
$$S_2 : b \rightarrow (\mathbf{1}' + i\,\mathbf{0}')/\sqrt{2} \quad , \tag{21}$$
$$S_2 : c \rightarrow (\mathbf{0}' + i\,\mathbf{1}')/\sqrt{2} \quad , \tag{22}$$
$$P_4 : \mathbf{0}' \rightarrow \mathbf{0}'e^{i\varphi} \quad . \tag{23}$$

When again $\mathbf{0} \equiv \mathbf{0}' \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\mathbf{1} \equiv \mathbf{1}' \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, then the corresponding unitary evolution matrix which transforms any coherent superposition of $\mathbf{0}$ and $\mathbf{1}$ into a superposition of $\mathbf{0}'$ and $\mathbf{1}'$ is given by

$$\mathbf{T}_{21}^{MZ}(\alpha,\beta,\omega,\varphi) = -i\,e^{-i(\beta+\frac{\omega}{2})} \begin{pmatrix} -e^{-i\,(\alpha+\varphi)}\sin\frac{\omega}{2} & e^{-i\,\varphi}\cos\frac{\omega}{2} \\ e^{-i\,\alpha}\cos\frac{\omega}{2} & \sin\frac{\omega}{2} \end{pmatrix} \quad . \tag{24}$$

The correspondence between $\mathbf{T}_{21}^{bs}(T(\omega),\alpha,\beta,\varphi)$ with $\mathbf{T}_{21}^{MZ}(\alpha',\beta',\omega',\varphi')$ in equations (15) (24) can be verified by comparing the elements of these matrices. The resulting four equations can be used to eliminate the four unknown parameters $\omega' = 2\omega$, $\beta' = \beta - \omega$, $\alpha' = \alpha - \pi/2$, $\beta' = \beta - \omega$ and $\varphi' = \varphi - \pi/2$; i.e.,

$$\mathbf{T}_{21}^{bs}(\omega,\alpha,\beta,\varphi) = \mathbf{T}_{21}^{MZ}(\alpha - \frac{\pi}{2}, \beta - \omega, 2\omega, \varphi - \frac{\pi}{2}) \quad . \tag{25}$$

Both elementary quantum interference devices are *universal* in the sense that *every* unitary quantum evolution operator in two-dimensional Hilbert space can be brought into a one-to-one correspondence to $\mathbf{T}_{21}^{bs}$ and $\mathbf{T}_{21}^{MZ}$; with corresponding values of T,α,β,φ or $\alpha,\omega,\beta,\varphi$. This can be easily seen by a similar calculation as before; i.e., by comparing equations (15) (24) with the "canonical" form of a unitary matrix, which is the product of a $U(1) = e^{-i\beta}$ and of the unimodular unitary matrix $SU(2)$ [25]

$$\mathbf{T}(\omega,\alpha,\varphi) = \begin{pmatrix} e^{i\alpha}\cos\omega & -e^{-i\varphi}\sin\omega \\ e^{i\varphi}\sin\omega & e^{-i\alpha}\cos\omega \end{pmatrix} \quad , \tag{26}$$

where $-\pi \leq \beta,\omega \leq \pi$, $-\frac{\pi}{2} \leq \alpha,\varphi \leq \frac{\pi}{2}$. Let

$$\mathbf{T}(\omega,\alpha,\beta,\varphi) = e^{-i\beta}\mathbf{T}(\omega,\alpha,\varphi) \quad . \tag{27}$$

A proper identification of the parameters $\alpha,\beta,\omega,\varphi$ yields

$$\mathbf{T}(\omega,\alpha,\beta,\varphi) = \mathbf{T}_{21}^{bs}(\omega - \frac{\pi}{2}, -\alpha - \varphi - \frac{\pi}{2}, \beta + \alpha + \frac{\pi}{2}, \varphi - \alpha + \frac{\pi}{2}) \quad . \tag{28}$$

Let us examine the realization of a few primitive logical "gates" corresponding to (unitary) unary operations on qubits. The "identity" element $\mathbf{I}$ is defined by $\mathbf{0} \to \mathbf{0}$, $\mathbf{1} \to \mathbf{1}$ and can be realized by

$$\mathbf{I} = T_{21}^{bs}(-\frac{\pi}{2}, -\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}) = T_{21}^{MZ}(-\pi,\pi,-\pi,0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad . \tag{29}$$

The "not" element is defined by $\mathbf{0} \to \mathbf{1}$, $\mathbf{1} \to \mathbf{0}$ and can be realized by

$$\mathtt{not} = T_{21}^{bs}(0,0,0,0) = T_{21}^{MZ}(-\frac{\pi}{2},0,0,-\frac{\pi}{2}) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad . \tag{30}$$

The next element, "$\sqrt{\mathtt{not}}$" is a truly quantum mechanical; i.e., nonclassical, one, since it converts a classical bit into a coherent superposition of $\mathbf{0}$

and $\mathbf{1}$. $\sqrt{\mathtt{not}}$ is defined by $\mathbf{0} \to \mathbf{0}+\mathbf{1}$, $\mathbf{1} \to -\mathbf{0}+\mathbf{1}$ and can be realized by

$$\sqrt{\mathtt{not}} = T_{21}^{bs}(-\frac{\pi}{4}, -\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}) = T_{21}^{MZ}(-\pi, \frac{3\pi}{4}, -\frac{\pi}{2}, 0) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} . \tag{31}$$

Note that $\sqrt{\mathtt{not}} \cdot \sqrt{\mathtt{not}} = \mathtt{not} \cdot \mathrm{diag}(1,-1) = \mathtt{not} \,(\mathrm{mod}\, 1)$. The relative phases in the output ports showing up in $\mathrm{diag}(1,-1)$ can be avoided by defining

$$\sqrt{\mathtt{not}}' = T_{21}^{bs}(-\frac{\pi}{4}, 0, \frac{\pi}{4}, 0) = T_{21}^{MZ}(-\frac{\pi}{2}, \frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}) = \frac{1}{2} \begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix} . \tag{32}$$

With this definition, $\sqrt{\mathtt{not}}'\sqrt{\mathtt{not}}' = \mathtt{not}$.

It is very important that the elementary quantum interference device realizes an arbitrary quantum time evolution of a two-dimensional system. The performance of the quantum interference device is determined by four parameters, corresponding to the phases $\alpha, \beta, \varphi, \omega$.

References

[1] Cristian Calude and F. Walter Meyerstein. Is the universe lawful? *Chaos, Solitons & Fractals*, 10(6):1075–1084, 1999.

[2] Karl Svozil. Quantum interfaces. e-print `arXiv:quant-ph/0001064` available http://arxiv.org/abs/quant-ph/0001064, 2000.

[3] John A. Wheeler. Law without law. In John A. Wheeler and W. H. Zurek, editors, *Quantum Theory and Measurement*, pages 182–213. Princeton University Press, Princeton, 1983. [48].

[4] Daniel M. Greenberger. Private communication.

[5] David Hilbert. Über das Unendliche. *Mathematische Annalen*, 95:161–190, 1926.

[6] Erwin Schrödinger. Die gegenwärtige Situation in der Quantenmechanik. *Naturwissenschaften*, 23:807–812, 823–828, 844–849, 1935. English translation in [70] and [48, pp. 152-167].

[7] Simon Kochen and Ernst P. Specker. The problem of hidden variables in quantum mechanics. *Journal of Mathematics and Mechanics*, 17(1):59 87, 1967. Reprinted in [71, pp. 235–263].

[8] W. K. Wooters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982.

[9] D. Dieks. Communication by EPR devices. *Physics Letters*, 92A(6):271–272, 1982.

[10] L. Mandel. Is a photon amplifier always polarization dependent? *Nature*, 304:188, 1983.

[11] Peter W. Milonni and M. L. Hardies. Photons cannot always be replicated. *Physics Letters*, 92A(7):321–322, 1982.

[12] R. J. Glauber. Amplifiers, attenuators and the quantum theory of measurement. In E. R. Pikes and S. Sarkar, editors, *Frontiers in Quantum Optics*. Adam Hilger, Bristol, 1986.

[13] C. M. Caves. Quantum limits on noise in linear amplifiers. *Physical Review*, D26:1817–1839, 1982.

[14] Simon Kochen and Ernst P. Specker. Logical structures arising in quantum theory. In *Symposium on the Theory of Models, Proceedings of the 1963 International Symposium at Berkeley*, pages 177–189, Amsterdam, 1965. North Holland. Reprinted in [71, pp. 209–221].

[15] Simon Kochen and Ernst P. Specker. The calculus of partial propositional functions. In *Proceedings of the 1964 International Congress for Logic, Methodology and Philosophy of Science, Jerusalem*, pages 45–57, Amsterdam, 1965. North Holland. Reprinted in [71, pp. 222–234].

[16] Garrett Birkhoff and John von Neumann. The logic of quantum mechanics. *Annals of Mathematics*, 37(4):823–843, 1936.

[17] Charles H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin. Experimental quantum cryptography. *Journal of Cryptology*, 5:3–28, 1992.

[18] Edward F. Moore. Gedanken-experiments on sequential machines. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*. Princeton University Press, Princeton, 1956.

[19] Cristian Calude, Elena Calude, Karl Svozil, and Sheng Yu. Physical versus computational complementarity I. *International Journal of Theoretical Physics*, 36(7):1495–1523, 1997.

[20] K. Svozil. *Quantum Logic.* Springer, Singapore, 1998.

[21] Artur Ekert. Quantum cryptography based on Bell's theorem. *Physical Review Letters*, 67:661–663, 1991.

[22] M. Hamrick G. Gilbert. Practical quantum cryptography: A comprehensive analysis (part one). MITRE report MTR 00W0000052 and e-print `arXiv:quant-ph/0009027` available at `http://arxiv.org/abs/quant-ph/0009027`, 2000.

[23] T. Jenewein, G. Weihs, C. Simon, H. Weinfurter, and A. Zeilinger. Poster, 1998.

[24] Karl Svozil. The information interpretation of quantum mechanics. e-print `arXiv:quant-ph/0006033` available http://arxiv.org/abs/quant-ph/0006033, 2000.

[25] F. D. Murnaghan. *The Unitary and Rotation Groups.* Spartan Books, Washington, 1962.

[26] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium of on Foundations of Computer Science, Santa Fe, NM, Nov. 20-22, 1994.* IEEE Computer Society Press, November 1994. `arXiv:quant-ph/9508027.`

[27] Artur Ekert and Richard Jozsa. Quantum computation and Shor's factoring algorithm. *Reviews of Modern Physics*, 68(3):733–753, 1996.

[28] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 212–219. 1996.

[29] Josef Gruska. *Quantum Computing.* McGraw-Hill, London, 1999.

[30] Georg Gottlob. Private communication.

[31] Cristian S. Calude, Michael J. Dinneen, and Karl Svozil. Reflections on quantum computing. *Complexity*, 2000; in print.`http://www.cs.auckland.ac.nz/CDMTCS//researchreports/130cris.pdf.`

[32] Max Planck. Ueber eine Verbesserung der Wien'schen Spectralgleichung. *Verhandlungen der deutschen physikalischen Gesellschaft*, 2:202, 1900. See also [33].

[33] Max Planck. Ueber das Gesetz der Energieverteilung im Normalspectrum. *Annalen der Physik*, 4:553–566, 1901.

[34] Max Planck. Zur Theorie des Gesetzes der Energieverteilung im Normalspectrum. *Verhandlungen der deutschen physikalischen Gesellschaft*, 2:237, 1900. See also [33].

[35] Albert Einstein. Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt. *Annalen der Physik*, 17:132–148, 1905.

[36] Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics. Quantum Mechanics*, volume III. Addison-Wesley, Reading, MA, 1965.

[37] E. G. Harris. *A Pedestrian Approach to Quantum Field Theory.* Wiley-Interscience, New York, 1971.

[38] H. J. Lipkin. *Quantum Mechanics, New Approaches to Selected Topics.* North-Holland, Amsterdam, 1973.

[39] L. E. Ballentine. *Quantum Mechanics.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[40] A. Messiah. *Quantum Mechanics*, volume I. North-Holland, Amsterdam, 1961.

[41] A. S. Davydov. *Quantum Mechanics.* Addison-Wesley, Reading, MA, 1965.

[42] P. A. M. Dirac. *The Principles of Quantum Mechanics.* Oxford University Press, Oxford, 1947.

[43] Asher Peres. *Quantum Theory: Concepts and Methods.* Kluwer Academic Publishers, Dordrecht, 1993.

[44] George W. Mackey. *The Mathematical Foundations of Quantum Mechanics.* W. A. Benjamin, Reading, MA, 1963.

[45] John von Neumann. *Mathematische Grundlagen der Quantenmechanik.* Springer, Berlin, 1932. English translation: *Mathematical Foundations of Quantum Mechanics*, Princeton University Press, Princeton, 1955.

[46] John S. Bell. *Speakable and Unspeakable in Quantum Mechanics.* Cambridge University Press, Cambridge, 1987.

[47] Max Jammer. *The Philosophy of Quantum Mechanics.* John Wiley & Sons, New York, 1974.

[48] John Archibald Wheeler and Wojciech Hubert Zurek. *Quantum Theory and Measurement.* Princeton University Press, Princeton, 1983.

[49] N. Dunford and J. T. Schwartz. *Linear Operators I.* Interscience Publishers, New York, 1958.

[50] Michael Reed and Barry Simon. *Methods of Mathematical Physics I: Functional Analysis.* Academic Press, New York, 1972.

[51] Michael Reed and Barry Simon. *Methods of Mathematical Physics II: Fourier Analysis, Self-Adjointness.* Academic Press, New York, 1975.

[52] M. Reck, Anton Zeilinger, H. J. Bernstein, and P. Bertani. Experimental realization of any discrete unitary operator. *Physical Review Letters*, 73:58 61, 1994. See also [25].

[53] Daniel B. Greenberger and A. YaSin. "Haunted" measurements in quantum theory. *Foundation of Physics*, 19(6):679-704, 1989.

[54] Thomas J. Herzog, Paul G. Kwiat, Harald Weinfurter, and Anton Zeilinger. Complementarity and the quantum eraser. *Physical Review Letters*, 75(17):3034–3037, 1995.

[55] David Finkelstein and Shlomit R. Finkelstein. Computational complementarity. *International Journal of Theoretical Physics*, 22(8):753–779, 1983.

[56] Gregory J. Chaitin. An improvement on a theorem by E. F. Moore. *IEEE Transactions on Electronic Computers*, EC-14:466–467, 1965.

[57] J. H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall Ltd., London, 1971.

[58] W. Brauer. *Automatentheorie.* Teubner, Stuttgart, 1984.

[59] Karl Svozil. *Randomness & Undecidability in Physics.* World Scientific, Singapore, 1993.

[60] Martin Schaller and Karl Svozil. Partition logics of automata. *Il Nuovo Cimento*, 109B:167-176, 1994.

[61] Martin Schaller and Karl Svozil. Automaton partition logic versus quantum logic. *International Journal of Theoretical Physics*, 34(8):1741–1750, August 1995.

[62] Martin Schaller and Karl Svozil. Automaton logic. *International Journal of Theoretical Physics*, 35(5):911 940, May 1996.

[63] Anatolij Dvurečenskij, Sylvia Pulmannová, and Karl Svozil. Partition logics, orthoalgebras and automata. *Helvetica Physica Acta*, 68:407–428, 1995.

[64] Karl Svozil and Roman R. Zapatrin. Empirical logic of finite automata: microstatements versus macrostatements. *International Journal of Theoretical Physics*, 35(7):1541–1548, 1996.

[65] Karl Svozil and Josef Tkadlec. Greechie diagrams, nonexistence of measures in quantum logics and Kochen–Specker type constructions. *Journal of Mathematical Physics*, 37(11):5380–5401, November 1996.

[66] M. Reck and Anton Zeilinger. Quantum phase tracing of correlated photons in optical multiports. In F. De Martini, G. Denardo, and Anton Zeilinger, editors, *Quantum Interferometry*, Singapore, 1994. World Scientific.

[67] Charles H. Bennett. Night thoughts, dark sight. *Nature*, 371:479–480, 1994.

[68] B. Yurke, S. L. McCall, and J. R. Klauder. SU(2) and SU(1,1) interferometers. *Physical Review*, A33:4033–4054, 1986.

[69] R. A. Campos, B. E. A. Saleh, and M. C. Teich. Fourth-order interference of joint single-photon wave packets in lossless optical systems. *Physical Review*, A42:4127, 1990.

[70] J. D. Trimmer. The present situation in quantum mechanics: a translation of Schrödinger's "cat paradox". *Proc. Am. Phil. Soc.*, 124:323-338, 1980. Reprinted in [48, pp. 152-167].

[71] Ernst Specker. *Selecta*. Birkhäuser Verlag, Basel, 1990.

TEIL B

Beiträge zum Theorietag

McNaughton Languages

M. Beaudry[1], M. Holzer[2] , G. Niemann[3] and F. Otto[3]

[1] Département de mathématique et informatique
Université de Sherbrooke
Sherbrooke, Québec, Canada J1K 2R1
beaudry@dmi.usherb.ca

[2] Département d'I.R.O.
Université de Montréal
Montréal, Québec, Canada H3C 3J7
holzer@iro.umontreal.ca

[3] Fachbereich Mathematik/Informatik
Universität Kassel, 34109 Kassel, Germany
{niemann,otto}@theory.informatik.uni-kassel.de

Abstract

By generalizing the Church-Rosser languages introduced by McNaughton et al in 1988 the concept of *McNaughton languages* is obtained. It is shown that this notion is as powerful as the notion of the Turing machine or the notion of the phrase-structure grammar. The various subclasses obtained by restricting the general model resemble several well-known complexity classes between the class CSL of context-sensitive languages and the recursively enumerable sets and build a hierarchy between the finite sets and CSL that refines the Chomsky hierarchy. The relationship between the various classes of McNaughton languages is investigated, and some closure and non-closure properties of these language classes as well as some complexity issues are considered.

Definitions and relations between the classes

We assume that the reader is familiar with the basic concepts of formal language and automata theory. As our standard reference concerning this

field we use the monographs by Hopcroft and Ullman [HU79] and Harrison [Har78]. In addition to this classical field of computer science we need some background from the theory of string-rewriting systems, where the monograph [BO93] serves as our main reference.

In [MNO88] the class CRL of *Church-Rosser languages* is introduced, where a language L is called a Church-Rosser language if it consists of those strings w that, placed into the context t_1wt_2 of certain auxiliary strings t_1 and t_2, reduce to a certain final symbol with respect to a finite, length-reducing, and confluent string-rewriting system. In this definition apart from the final symbol and the symbols occurring in the context strings t_1 and t_2, also other non-terminal symbols are allowed, e.g., for marking the left and the right border of the input strings.

Here we generalize this concept by admitting other classes of finite string-rewriting systems in the definition. This leads to the concept of *McNaughton languages*, where we denote the most general class by McNL.

Definition 1. *A language $L \subseteq \Sigma^*$ is called a McNaughton language if there exist a finite alphabet Γ strictly containing Σ, a finite string-rewriting system R on Γ, strings $t_1, t_2 \in (\Gamma \setminus \Sigma)^* \cap \mathrm{IRR}(R)$, and a letter $Y \in (\Gamma \setminus \Sigma) \cap \mathrm{IRR}(R)$ such that, for all $w \in \Sigma^*$, the following two statements are equivalent:*

(1.) $w \in L$;

(2.) $t_1wt_2 \rightarrow_R^* Y$.

Here the symbols of Σ are terminals, while those of $\Gamma \setminus \Sigma$ can be seen as non-terminals.

By placing restrictions on the finite string-rewriting systems used we obtain certain subclasses of the class McNL of all McNaughton languages. The above notion is as powerful as the notion of the Turing machine or the notion of the phrase-structure grammar. In fact for sufficiently large complexity classes $\mathcal{C}$ by restricting the derivation length of the string-rewriting systems used through a function from $\mathcal{C}$ we obtain the complexity class $\mathcal{C}$. However, for more restricted string-rewriting systems we obtain language classes inside the class CSL of context-sensitive languages, which do not seem to coincide with any well-known language classes. A special case, for example, is the class CRL of Church-Rosser languages [MNO88].

A McNaughton language will be called *confluent* if it is specified by a finite confluent string-rewriting system. By con-McNL we denote the class of confluent McNaughton languages. Next we introduce some classes of McNaughton languages obtained by several restrictions on the string-rewriting systems used in the specification.

f-b-McNL: *f-bounded McNaughton languages* for a function $f : \mathbb{N} \to \mathbb{N}$ are defined using a finite terminating string-rewriting system R such that for each $n \in \mathbb{N}$ the longest reduction in R starting with a word of length n has at most $f(n)$ many steps.

$\mathcal{C}$-b-McNL: *$\mathcal{C}$-bounded McNaughton languages* for a class of functions are defined using a finite terminating string-rewriting system R which is f-bounded for a function $f \in \mathcal{C}$.

prim-rec-McNL: denotes the class of McNaughton languages that are bounded by a primitive-recursive function.

non-in-McNL: *non-increasing McNaughton languages* are defined using a finite terminating string-rewriting system R such that $|\ell| \geq |r|$ holds for each rule $(\ell \to r) \in R$.

lr-McNL: *length-reducing McNaughton languages* are defined using a finite string-rewriting system R that is length-reducing, that is, for each rule $(\ell \to r) \in R$ we have $|\ell| > |r|$.

non-win-McNL: *non-weight-increasing McNaughton languages* are defined using a finite terminating string-rewriting system R such that there exists a weight-function φ satisfying $\varphi(\ell) \geq \varphi(r)$ for each rule $(\ell \to r) \in R$. Here a weight-function is a morphism $\varphi : \Sigma^* \to \mathbb{N}_+$.

wr-McNL: *weight-reducing McNaughton languages* are defined using a weight-reducing string-rewriting system R, that is, there exists a *weight-function* $\varphi : \Sigma^* \to \mathbb{N}_+$ satisfying $\varphi(\ell) > \varphi(r)$ for each rule $(\ell \to r) \in R$.

mon-McNL: *monadic McNaughton languages* are defined using a monadic string-rewriting system R, that is, R is length-reducing and for each rule $(\ell \to r)$ of R holds $|r| \leq 1$.

sp-McNL: *special McNaughton languages* are defined using a special string-rewriting system R, that is, R is length-reducing and the right-hand side of each rule is the empty string.

The confluent counterparts of these classes are denoted with the prefix con.

Last we introduce still another class of languages, which can be seen as a variant of the class CRL of Church-Rosser languages without auxiliary symbols and without context strings.

Definition 2. *A language $L \subseteq \Sigma^*$ is called a Church-Rosser congruential language if there exist a finite, length-reducing, and confluent string-rewriting system R on Σ and finitely many irreducible strings $w_1, w_2, \ldots, w_n \in \Sigma^*$ such that $L = \bigcup_{i=1}^{n} [w_i]_R$.*

The hierarchy of classes of McNaughton languages is depicted in Figure 1, where the relations concernig CRCL follow from [MNO88]. Here RE is the class of all recursively enumerable languages, GCSL is the class of growing context-sensitive languages [DW86], and FIN denotes the class of all finite languages, E_n denotes the n-th level of the Grzegorczyk hierarchy [Grz53, Tou84]. The class of primitive-recursive functions coincides with the union $\bigcup_{n \geq 0} \mathsf{E}_n$. In particular, it is also a Turing machine time-complexity class. It is known that a function f belongs to the Grzegorczyk class E_2 if and only if is computed by a deterministic Turing machine in polynomial time.

All inclusions are known to be proper apart from the well-known LBA- and P-NP-problems. At this time it remains open whether or not sp-McNL is contained in CRL. Also it is a long-standing open problem whether or not each regular language is in CRCL. For a partial result showing that at least all regular languages of polynomial density are in CRCL see [Nie00]. All other inclusions that are not depicted do not hold.

The result concernig E_2-bounded McNaughton languages very nicely illustrates the fact that confluent McNaughton classes correspond to deterministic complexity or language classes, while non-confluent McNaughton classes correspond to non-deterministic complexity or language classes.

Closure properties and complexity issues

From the definition it is immediate that each class of McNaughton languages is closed under reversal and under quotient with a single string.

Other closure properties of the various classes of McNaughton languages can be seen from the following table.

	$\cup$	$\cap$	co	$\cdot$	$\cap$REG	inv. hom.	ε-free hom.	hom.
con-sp-McNL	–	–	–	–	–	?	?	?
sp-McNL	–	–	–	–	–	+	+	+
con-mon-McNL	–	–	?	–	?	?	?	?
mon-McNL=CFL	+	–	–	+	+	+	+	+
con-lr-McNL=CRL	–	–	+	–	+	+	–	–
lr-McNL=GCSL	+	–	–	+	+	+	+	–

In [Her97] Herzog establishes hierarchies of languages that are obtained as unions of bounded numbers of deterministic context-free languages, where the levels of this strict hierarchy resemble the degree of nondeterminism and ambiguity in a context-free language.

Analoguously hierarchies of languages that are obtained as unions of bounded numbers of confluent and monadic, respectively special, McNaughton languages can be established. With respect to the degree of nondeterminism and the degree of ambiguity these classes yield the same strict hierarchies as the corresponding classes built on DCFL. For details see [BHNO00].

Finally we turn to the task of determining the degree of complexity of the fixed and the general membership problems for those McNaughton languages that are specified by monadic or special string-rewriting systems. It can be seen from the following table.

	fixed membership	general membership
mon-McNL	LOG(CFL)-complete	LOG(CFL)-complete
con-mon-McNL	$\subseteq$ LOG(DCFL) L-hard under $\leq^m_{\mathsf{NC}^1}$	$\subseteq$ LOG(DCFL) L-hard under $\leq^m_{\mathsf{NC}^1}$
sp-McNL	LOG(CFL)-complete	LOG(CFL)-complete
con-sp-McNL	$\subseteq$ LOG(DCFL) NC^1-hard under $\leq^m_{\mathsf{AC}^0}$	$\subseteq$ LOG(DCFL) NC^1-hard under $\leq^m_{\mathsf{AC}^0}$

Open Questions

Here is a short list containing the open problems that we did encounter before:

1. Is the class sp-McNL contained in CRL?
2. Is REG contained in CRCL?
3. Is the class con-mon-McNL closed under complementation or intersectiony with regular sets?
4. Are the fixed and the general membership problems for con-sp-McNL LOG(DCFL)-hard?
5. Are these problems for the class con-mon-McNL LOG(DCFL)-hard?

As neither the class con-sp-McNL nor the class sp-McNL contain all the regular sets, it is certainly of interest to investigate the properties of the intersections REG$\cap$con-sp-McNL and REG$\cap$sp-McNL. What closure properties

do they have? How can the languages in these intersections be characterized?

The various classes of McNaughton languages have been obtained as a generalization of the Church-Rosser languages. The Church-Rosser congruential languages can be seen as the *pure* variants of the Church-Rosser languages. Accordingly, this notion can also be generalized similarly, thus defining a hierarchy of *McNaughton congruential languages.*

Also instead of only distinguishing between confluent and non-confluent string-rewriting systems we could consider various classes of *weakly confluent* systems [MNOZ93], leading to a family of *weakly confluent McNaughton languages.*

References

[BHNO00] Martin Beaudry, Markus Holzer, Gundula Niemann, and Friedrich Otto. McNaughton Languages. in preparation, 2000.

[BO93] Ronald V. Book and Friedrich Otto. *String-Rewriting Systems.* Texts and Monographs in Computer Science. Springer, New-York, 1993.

[DW86] Elias Dahlhaus and Manfred K. Warmuth. Membership for growing context–sensitive grammars is polynomial. *Journal of Computer and System Sciences*, 33:456–472, 1986.

[Grz53] A. Grzegorczyk. Some classes of recursive functions. *Rozprawy Matematycne*, 4:1 45, 1953.

[Har78] Michael A. Harrison. *Introduction to Formal Language Theory.* Series in Computer Science. Addison-Wesley, Reading, MA, 1978.

[Her97] Christian Herzog. Pushdown automata with bounded nondeterminism and bounded ambiguity. *Theoretical Computer Science*, 181:141–157, 1997.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Series in Computer Science. Addison-Wesley, Reading, MA, 1979.

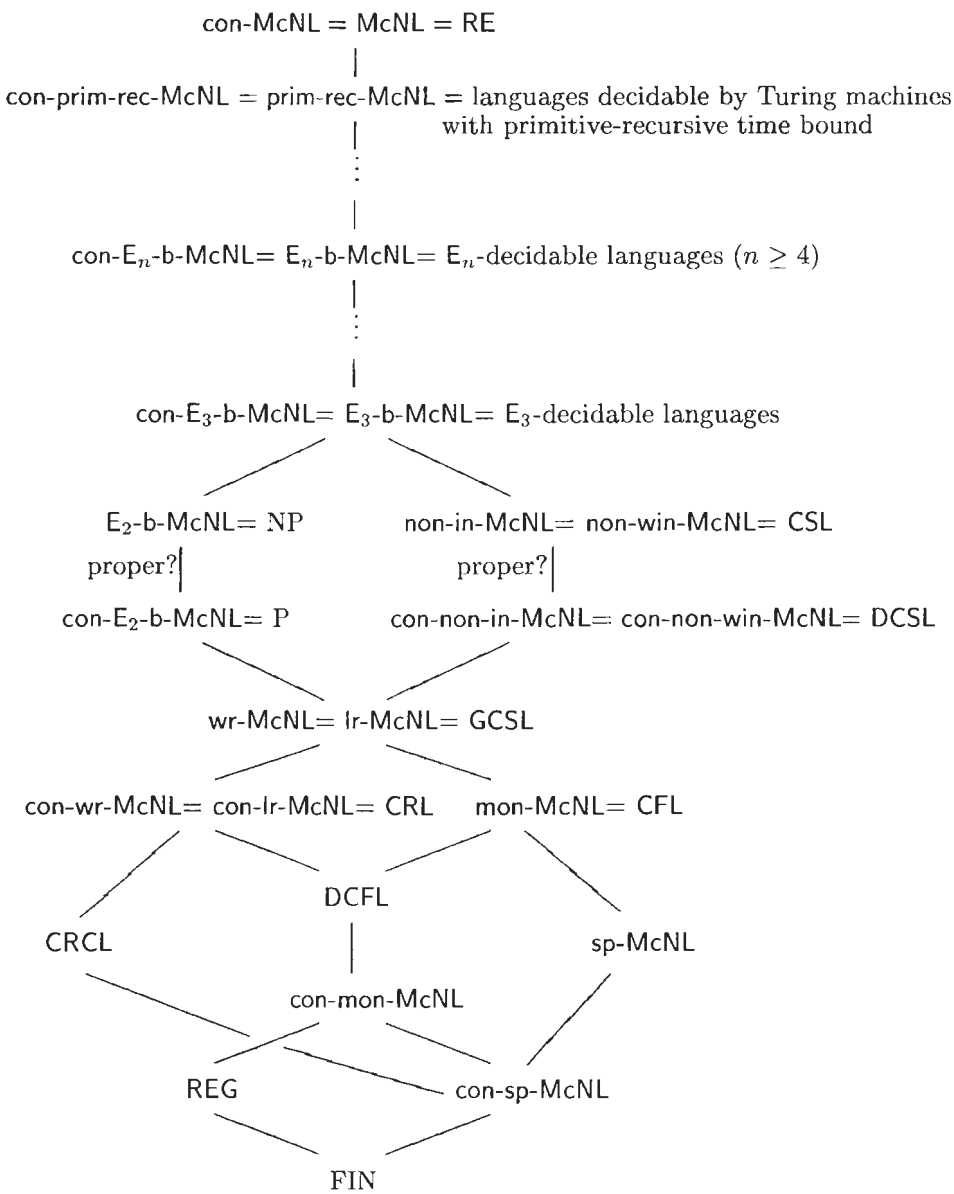
[MNO88] Robert McNaughton, Paliath Narendran, and Friedrich Otto. Church-Rosser Thue systems and formal languages. *Journal of the Association Computing Machinery*, 35:324–344, 1988.

[MNOZ93] K. Madlener, P. Narendran, F. Otto, and L. Zhang. On weakly confluent monadic string-rewriting systems. *Theoretical Computer Science*, 113:119–165, 1993.

[Nie00] Gundula Niemann. Regular Languages and Church-Rosser Congruential Languages.. In Rudolf Freund and Alica Kelemenova, editors, *Proceedings of the International Workshop Grammar Systems 2000*, pages 359 370. Silesian University at Opava, Faculty of Philosophy and Science, Institute of Computer Science, 2000.

[Tou84] G.J. Tourlakis. *Computability.* Reston, 1984.

Figure 1: The hierarchy of McNaughton languages



Deciding LTL over Mazurkiewicz Traces

Benedikt Bollig[1] and Jesper Henriksen[2] and Martin Leucker[1]

[1] Lehrstuhl für Informatik II
Aachen University of Technology, Germany
{bollig, leucker}@informatik.rwth-aachen.de
[2] **BRICS**, Department of Computer Science,
University of Aarhus, Denmark
gulmann@brics.dk

1 Introduction

A traditional approach towards automatic program verification is model checking specifications in linear time temporal logic [MP92]. The automata-theoretic approach has proven to be very simple and useful in solving this problem. Efficient (on–the fly) algorithms for checking satisfiability of such specifications are then derived by constructing (e.g. Büchi, Streett or alternating) automata accepting the set of sequences satisfying the formula at hand and checking the corresponding automaton for emptiness.

Systems are often distributed in nature and consist of components with a fixed static notion of independence working together concurrently. Traditionally, these are transformed into a single system by *interleaving*. However, this causes an exponential blow up in the number of components, known as *state space explosion*. Besides *symbolic model checking* [McM93], *partial order reduction* [Pel98] is a powerful approach to avoid this explosion. Its idea is to simplify the resulting system by employing its inherent independence. Of course, partial order reduction is only applicable for specifications that respect this independence. However, to decide whether a formula is so–called *trace consistent* is complete in PSPACE [Pel98].

An approach on the same line is to describe the runs of a system by partial orders which are *Mazurkiewicz traces* [Maz77, DR95] and to formulate specifications directly for traces. This guarantees that every formula of a corresponding logic is automatically trace consistent. Furthermore, given a decision procedure in terms of an automaton accepting all linearisations of

the partial orders satisfying the specification, well understood partial order reduction techniques can be employed and will yield considerable results.

Recently, a linear time temporal logic (LTL) over traces was introduced by Diekert and Gastin [DG00] and shown to be expressive complete wrt. first order logic (FO) over traces. Recalling Kamp's [Kam68] important result that LTL over words is expressive complete wrt. FO over words, we learn that LTL (over traces) covers exactly those specifications expressible in LTL (over words) which are trace consistent.

LTL is a simplification of LTrL [TW97], the first linear time temporal logic over traces shown to expressive complete (wrt. FO over traces). LTL coincides with LTrL by syntax and semantics of all operators except that it avoids a past tense modality. LTrL turned out to be non-elementary due to the *until* operator [Wal98]. As LTL coincides with LTrL wrt. the semantics of the *until*-operator, it is non elementary as well. Although this might limit LTL's practical applicability on the first sight, one should consider that typical specifications only have a small number of nested *until*-formulas. Hence, a decision procedure for LTL should work in practice very well even if its worst-case complexity is high. Similar experiences were made with second order logic over words [HJJK95].

A decision procedure for LTrL was presented in [GMP98], which can easily be adapted for LTL. The construction of a Büchi automaton is carried out according to the inductive definition of a formula. For example, for a formula φ and its corresponding Büchi automaton $\mathcal{A}_\varphi$, the one for $\neg\varphi$ is obtained by complementing $\mathcal{A}_\varphi$. This implies a non-elementary blow-up also with respect to negations which limits its practical usage. For model checking, where typically an automaton corresponding to a formula is combined with one describing an underlying system, the construction has another disadvantage. It works in a *bottom-up* manner and implies that the whole automaton has to be constructed before a verification process can be started. Therefore it is not applicable for *on-the-fly* model checking.

Our procedure is based on alternating automata (see [Var96]) and has the advantage that an exponential blow-up only occurs for nested *untils*, which cannot be avoided. Furthermore, our automaton can be constructed *on-the-fly*, i.e., only the part of the automaton corresponding to the part of the formula currently considered has to be constructed.

Beyond the decision procedure itself, our approach shows that the elegant alternating automata approach can be carried over to traces successfully.

A full version of the paper (including proofs) can be obtained by contacting the authors.

Acknowledgement: Part of the work was done during the last author's

stay at BRICS. He is grateful for the hospitality and the overall support.

2 Preliminaries

A *(Mazurkiewicz) trace alphabet* is a pair (Σ, I), where Σ, the alphabet, is a finite set and $I \subseteq \Sigma \times \Sigma$ is an irreflexive and symmetric *independence relation*. Usually, Σ consists of the actions performed by a distributed system while I captures a static notion of causal independence between actions. For the rest of the section, we fix a trace alphabet (Σ, I). We define $D = (\Sigma \times \Sigma) - I$ to be the *dependency relation*, which is then reflexive and symmetric. Instead of $(a, b) \in I$, we simply write aIb. For sets $X, Y \subseteq \Sigma$, XIY denotes that all actions of X and Y are pairwise independent, and XDY denotes that there is an action $a \in X$ and $b \in Y$ such that a and b are dependent. If $X = \{a\}$ is a singleton, we omit the curly braces and write aIY or YIa.

Let $T = (E, \leq, \lambda)$ be a Σ-labelled poset, i.e., $(E, \leq)$ is a poset and $\lambda : E \to \Sigma$ is a labelling function. λ can be extended to subsets of E in the expected manner. For $e \in E$, we define its *history* as $\downarrow e = \{x \in E \mid x \leq e\}$. We also let $\lessdot$ be the *covering relation* given by $x \lessdot y$ iff $x < y$ and for all $z \in E$, $x \leq z \leq y$ implies $x = z$ or $z = y$.

A *(Mazurkiewicz) trace* (over (Σ, I)) is a Σ-labelled poset $T = (E, \leq, \lambda)$ such that, for all $e, e' \in E$, $\downarrow e$ is a finite set, $e \lessdot e'$ implies $\lambda(e)\ D\ \lambda(e')$, and $\lambda(e)\ D\ \lambda(e')$ implies $e \leq e'$ or $e' \leq e$.

We shall let $TR(\Sigma, I)$ denote the class of traces over (Σ, I). As usual, a trace language L is a subset of traces, i.e. $L \subseteq TR(\Sigma, I)$. Throughout the paper, we will not distinguish between isomorphic elements in $TR(\Sigma, I)$. We will refer to members of E as *events*.

Let $T = (E, \leq, \lambda)$ be a trace over (Σ, I). A *configuration* of T is a finite subset of events $c \subseteq E$ with $\downarrow c = c$, where $\downarrow c = \bigcup_{e \in c} \downarrow e$. The set of configurations of T will be denoted by $\mathcal{C}_T$. Trivially, $\emptyset \in \mathcal{C}_T$ is always the case.

Moreover, $\mathcal{C}_T$ can be equipped with a natural transition relation $\longrightarrow_T \subseteq \mathcal{C}_T \times \Sigma \times \mathcal{C}_T$ given by $c \xrightarrow{a}_T c'$ iff there exists an $e \in E$ such that $\lambda(e) = a$, $e \notin c$, and $c' = c \cup \{e\}$. The formulas of the temporal logics of traces are interpreted at configurations of traces.

An *ω-linearisation* of a trace $(E, \leq, \lambda)$ is a labelled total order $(E, \leq', \lambda)$ such that $\leq \subseteq \leq'$ and every event $e \in E$ has a finite history wrt. $\leq'$, i.e., $\forall e \in E\ |\downarrow e| < \infty$. An ω-linearisation of a trace corresponds to an ω-word, and all ω-linearisations of a trace form an equivalence class in Σ^ω. Conversely, every ω-word w denotes a unique trace T denoted by $str(w)$,

and every prefix v of w yields a configuration $\rho(v)$ of $str(w)$.

3 Alternating Büchi Automata

Let us recall the notion of alternating Büchi Automata along the lines of [Var96], which may also be consulted for a detailed introduction. A thorough investigation on Büchi automata can be found in [Tho90].

For a finite set X of variables, let $\mathcal{B}^+(X)$ be the set of *positive Boolean formulas* over X, i.e., the smallest set such that $X \subseteq \mathcal{B}^+(X)$, $\mathtt{true}, \mathtt{false} \in \mathcal{B}^+(X)$, and $\mathcal{B}^+(X)$ also contains $\varphi \wedge \psi$ as well as $\varphi \vee \psi$ for $\varphi, \psi \in \mathcal{B}^+(X)$.

We say that a set $Y \subseteq X$ *satisfies* a formula $\varphi \in \mathcal{B}^+(X)$ ($Y \models \varphi$) iff φ evaluates to *true* when the variables in Y are assigned to *true* and the members of $X \backslash Y$ are assigned to *false*. For example, $\{q_1, q_3\}$ as well as $\{q_1, q_4\}$ satisfy the formula $(q_1 \vee q_2) \wedge (q_3 \vee q_4)$.

An *Alternating Büchi Automaton* (ABA) over Σ is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \mathcal{F})$ such that Q is a finite nonempty set of *states* , $q_0 \in Q$ is the *initial state* , $\mathcal{F} \subseteq Q$ is a set of accepting states, and $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is the *transition function* .

In a quite informal, intuitive manner, a run of $\mathcal{A}$ on a word $w = \alpha(0)\alpha(1)\ldots \in \Sigma^\omega$ is a pair (t, T) where t is a tree and $T : nodes(t) \to Q$ a labelling function such that the root of t is labelled by q_0 and for every node $n \in nodes(t)$, it holds either both $\delta(T(n), \alpha(height(n))) \in \{\mathtt{true}, \mathtt{false}\}$ and n has no children or, otherwise, $\{T(m) \mid m \in children(n)\} \models \delta(T(n), \alpha(height(n)))$. Let thereby $height(n) = 0$ if n is root, $height(n) = height(father(n)) + 1$, otherwise. The run τ is *accepting* if for every leaf node n, it holds $\delta(T(n), \alpha(height(n))) = \mathtt{true}$ and every infinite branch of τ hits an element of $\mathcal{F}$ infinitely often. The language $\mathcal{L}(\mathcal{A})$ of an automaton $\mathcal{A}$ is determined by all words for which an accepting run of $\mathcal{A}$ exists.

Every Büchi automaton can easily be turned into an equivalent (wrt. the accepted language) alternating Büchi automaton of the same size. The converse is also true [Var96], but the construction involves an exponential blow up. Hence, it is easy to see that the emptiness problem for ABAs is exponential in the number of states.

4 Deciding LTL

Let (Σ, I) be a trace alphabet. The formulas of LTL(Σ, I) are then given by the smallest set that for all $\varphi, \psi \in$ LTL(Σ, I) satisfies: $\mathrm{tt}, \neg\varphi, \varphi \vee \psi, \langle a \rangle \varphi, \varphi \mathcal{U} \psi \in$ LTL(Σ, I). Given a trace $T \in TR(\Sigma, I)$, $c \in \mathcal{C}_T$, and a

formula $\varphi \in \mathrm{LTL}(\Sigma, I)$, the notion of $T, c \models \varphi$ is defined inductively via (tt, negation, and disjunction interpreted as usual):

- $T, c \models \langle a \rangle \varphi$ iff there exists a $c' \in \mathcal{C}_T$ such that $c \xrightarrow{a}_T c'$ and $T, c' \models \varphi$.
- $T, c \models \varphi \mathcal{U} \psi$ iff there exists a $c' \in \mathcal{C}_T$ with $c \subseteq c'$ such that $T, c' \models \psi$ and all $c'' \in \mathcal{C}_T$ with $c \subseteq c'' \subset c'$ satisfy φ.

Note that LTL can be enriched by operators $\wedge$, $\Diamond$, and $\Box$ for expressing conjunctions, eventually, and globally which can easily be defined as abbreviations.

In the construction, we will extend LTL with an indexed until operator $\Phi \mathcal{U}^Z \psi$ where $\Phi = \{\varphi_1^{Y_1}, \ldots, \varphi_N^{Y_N}\}$ is a set of indexed formulas. It has the following semantics:

- $T, c \models \Phi \mathcal{U}^Z \psi$ iff there exists a $c' \in \mathcal{C}_T$ with $c \subseteq c'$ such that $T, c' \models \psi$ and $\lambda(c' - c) I Z$ and for all $i \in \{1, \ldots, N\}$ and all c'' with $c \subseteq c'' \subset c'$ and $\lambda(c'' - c) I Y_i$, it holds $c'' \models \varphi_i$.

Note that $\varphi \mathcal{U} \psi$ can be identified with $\{\varphi^{\emptyset}\} \mathcal{U}^{\emptyset} \psi$. We now describe the construction of an alternating Büchi automaton accepting all linearisations of traces satisfying the underlying formula of LTL.

Our approach can be understood as a kind of *hierarchical automaton*. Let us assume that the actions a and b are independent, and let us consider the formula $\eta = \langle a \rangle \varphi$. Suppose our automaton is in the state η and reads an a. Then it is natural to proceed in the state φ, since the required a has been seen. Reading a b instead, however, must not yield the state *false* since b is independent of a. Instead, we consider the resulting state ψ of an automaton starting in φ and reading the b. This ψ is now preceded with $\langle a \rangle$, yielding the state $\langle a \rangle \psi$.

This means that the state space of our automaton will consist of any derivative of our formula η which can be obtained by substituting any of its subformulas by a positive boolean combination of the corresponding subformula. The motivation for the *until*–operator is given in the next subsection. Let us be more precise:

Definition 4.1 (Extended-Closure) *For $\eta \in \mathrm{LTL}(\Sigma, I)$, let $ecl(\eta)$ be the smallest set that contains all formulas achieved by removing arbitrary occurences of operators $\langle a \rangle$ from η and that is closed wrt. subformulas in the usual sense and under positive Boolean combination. Furthermore, it holds $\eta = \varphi \mathcal{U} \psi \Rightarrow \forall Z \subseteq \Sigma\ \mathcal{P}(ecl(\varphi)^{\mathcal{P}(Y)}) \mathcal{U}^Z ecl(\psi) \subseteq ecl(\eta)$. In addition, every subformula can be replaced by an arbitrary element of its extended closure.*

Prop 4.2 *For $\eta \in \mathrm{LTL}(\Sigma, I)$ $|ecl(\eta)|$ is non-elementary.*

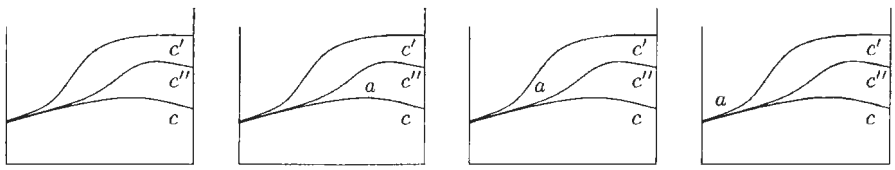


Figure 1: Configuration and actions

Rewriting the formula We now introduce the operator *rew*, which will be used in our construction for manipulating the underlying formula. For example, tt is rewritten by any action to tt. Rewriting $\langle b\rangle\varphi$ by a yields φ, if $a = b$, or ff, if $a \neq b$ and aDb. In the case that a and b are independent, we rewrite φ by a yielding φ' and let the result of $rew(\langle b\rangle\varphi)$ be $\langle b\rangle\varphi'$.

Before we give a precise definition of our rewriting operator, we analyse the semantics of the *until* graphically. Suppose we have the following trace and consider the configuration c and a future configuration c', i.e., $c' \supseteq c$. Furthermore, let c'' be a configuration between c and c' (Figure 1 (i)).

Suppose we can augment c by an event e labelled by a to obtain a successor configuration c''' of c. Then $c''' \subseteq c'' \subseteq c'$ or $c''' \not\subseteq c''$ but $c''' \subseteq c'$ or $c''' \not\subseteq c'$ which is shown in Figure 1 (a) – (c). In Case (b), it is obvious that $\lambda(c''-c)Ia$ and for Case (c), we have $\lambda(c'-c)Ia$ (as well as $\lambda(c''-c)Ia$).

The situation shown in Figure 1 can be described in other words in the following way: The action a is neither in the future of c'' nor of c' (Case (a)), just in the future of c'' (Case (b)), or in the future of c'' as well as of c' (Case (c)).

Let us consider a formula $\varphi\mathcal{U}\psi$ to be checked in the configuration c. In Case (c), we have to employ a for verifying ψ as well as φ. Note, for (c) we get two subcases, $c' = c$ or $c' \supset c$. While in the first case, we don't have to care about φ, we have to prove φ in the configurations between c and c'. Note that these configurations are reached by actions independent of a.

For Case (a), we have to employ the a for verifying φ in configuration c but not in c''. In Case (b), we have to prove φ considering a in the configuration c'' which might be equal to c as well as different from c. Note that in the latter case, every event of $c'' - c$ is independent of a.

Consequently, we define the rewriting operator for a formula $\Phi\mathcal{U}^Z\psi$ as follows. Let $\Psi_1 = rew(\psi, a)$, $\Psi_2 = \{rew(\varphi, a)^{Y\cup\{a\}} \mid \varphi^Y \in \Phi\}\mathcal{U}^{Z\cup\{a\}} rew(\psi, a)$, and $\Psi' = \Psi_1 \vee \Psi_2$. Let $rew(\Phi, a) = \{rew(\varphi, a)^{Y\cup\{a\}} \mid \varphi^Y \in \Phi\} \cup \{\varphi^Y \mid \varphi^Y \in \Phi, aIY\}$ and $\Phi_1 = \bigwedge_{\varphi^Y\in\Phi} rew(\varphi, a)$ $\Phi_2 =$

$rew(\Phi, a)\mathcal{U}^Z\psi$ and $\Phi' = \Phi_1 \wedge \Phi_2$. We define

$$rew(\Phi\mathcal{U}^Z\psi, a) = \begin{cases} \Psi' & \text{if } aDZ \\ \Psi' \vee \Phi' & \text{if } aIZ \end{cases}$$

Ψ' captures Case (c) in which an action a is employed for verifying ψ under the assumption that $c' = c$ (Ψ_1) or not (Ψ_2). Φ' covers the idea that a is not in the future of c' but is employed for verifying the obligations in Φ.

Altogether, the formal definition of the rewrite operator is as follows:

Definition 4.3 *For $\eta \in \mathrm{LTL}(\Sigma, I)$ we define the operator $rew : ecl(\eta) \times \Sigma \to \mathcal{B}^+(ecl(\eta))$[1] by*

$$\begin{array}{rcl} (\mathrm{tt}, a) & \mapsto & \mathrm{tt} \\ (\varphi \vee \psi, a) & \mapsto & rew(\varphi, a) \vee rew(\psi, a) \\ (\neg\varphi, a) & \mapsto & \overline{rew(\varphi, a)} \\ (\langle b\rangle\varphi, a) & \mapsto & \begin{cases} \varphi & \text{if } a = b \\ \langle b\rangle rew(\varphi, a) & \text{if } aIb \\ \mathrm{ff} & \text{if } aDb, a \neq b \end{cases} \\ (\varphi\mathcal{U}\psi, a) & \mapsto & rew(\{\varphi^\emptyset\}\mathcal{U}^\emptyset\psi, a) \\ (\Phi\mathcal{U}^Z\psi, a) & \mapsto & = \begin{cases} \Psi' & \text{if } aDZ \\ \Psi' \vee \Phi' & \text{if } aIZ \end{cases} \end{array}$$

$\overline{\varphi}$ denotes the dual of a formula φ, which is obtained as usual *viz* by applying de Morgan's laws to push down negation as far as possible. For example, the dual of the formula $\neg(\langle a\rangle\mathrm{tt} \vee \mathrm{tt}\mathcal{U}\langle b\rangle\mathrm{tt})$ is $\neg\langle a\rangle\mathrm{tt} \wedge \neg(\mathrm{tt}\mathcal{U}\langle b\rangle\mathrm{tt})$.

To show the correctness of our construction, the following proposition is essential.

Prop 4.4 *Let $\eta \in \mathrm{LTL}(\Sigma, I)$. For every $w \in \Sigma^\omega$ and every decomposition of w into $w = vaw'$, $v \in \Sigma^*$, we have $str(w), \rho(v) \models \eta \Leftrightarrow str(w), \rho(va) \models rew(\eta, a)$.*

The automaton The operators introduced in the preceding subsections allow a simple definition of the automaton accepting all the linearisations of a trace satisfying the underlying formula. We then maintain the correctness of our construction (Theorem 4.6).

Definition 4.5 *For $\varphi \in \mathrm{LTL}(\Sigma, I)$, the alternating Büchi automaton $\mathcal{A}_\varphi$ is the tuple $(Q, \Sigma, \delta, q_0, F)$ where $Q = ecl(\varphi)$ is the set of states, $\delta = rew$ is the transition function, $q_0 = \varphi$ is the initial state, and $F = \{\neg(\Phi\mathcal{U}^Z\psi) : \neg(\Phi\mathcal{U}^Z\psi) \in ecl(\varphi)\}$ is the set of accepting states.*

[1] Since $ecl(\eta)$ is closed under positive boolean combination we have $ecl(\eta) = \mathcal{B}^+(ecl(\eta))$

We conclude by formulating our main theorem:

Theorem 4.6 *Given a formula* $\varphi \in \mathrm{LTL}(\Sigma, I)$ *and its alternating Büchi automaton* $\mathcal{A}_\varphi = (Q, \Sigma, \delta, q_0, F)$, *for all* $w \in \Sigma^\omega$, *it holds* $w \in \mathcal{L}(\mathcal{A}_\varphi) \Leftrightarrow str(w), \emptyset \models \varphi$.

References

[DG00] Volker Diekert and Paul Gastin. Ltl is expressively complete for mazurkiewicz traces. In *Proceedings of 27th International Colloquium on Automata, Languages and Programming ICALP'2000*, volume 1853 of *LNCS*, Springer-Verlag, 2000.

[DR95] Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces.* World Scientific, Singapore, 1995.

[GMP98] P. Gastin, R. Meyer, and A. Petit. A (non-elementary) modular decision procedure for LTrL. Technical report, LSV, ENS de Cachan, 1998. extended version of MFCS'98.

[IIJJK95] J. G. Henriksen, J. Jensen, M. Joergensen, and N. Klarlund. MONA: Monadic second-order logic in practice. *Lecture Notes in Computer Science*, 1019:89–??, 1995.

[Kam68] H. W. Kamp. *Tense Logic and the Theory of Linear Order.* PhD thesis, University of California, Los Angeles, 1968.

[Maz77] Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.

[McM93] K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, Norwell Massachusetts, 1993.

[MP92] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer, New York, 1992.

[Pel98] Doron Peled. Ten years of partial order reduction. In *CAV, Computer Aided Verification*, number 1427 in LNCS, pages 17–28, Vancouver, BC, Canada, 1998. Springer.

[Tho90] Wolfgang Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science Publishers B. V., 1990.

[TW97] P. S. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for Mazurkiewicz traces. In *Proceedings, Twelth Annual IEEE Symposium on Logic in Computer Science*, pages 183–194, Warsaw, Poland, 29 June–2 July 1997. IEEE Computer Society Press.

[Var96] Moshe Y. Vardi. *An Automata-Theoretic Approach to Linear Temporal Logic*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag Inc., New York, NY, USA, 1996.

[Wal98] Igor Walukiewicz. Difficult configurations - on the complexity of LTrL. In *ICALP '98*, volume 1443 of *LNCS*, pages 140–151, 1998.

On the Number of Active Symbols in L and CD Grammar Systems

Henning Bordihn
Fakultät für Informatik, Otto-von-Guericke-Universität,
Postfach 4120, D-39016 Magdeburg, Germany
email: bordihn@iws.cs.uni-magdeburg.de

Markus Holzer*
Institut für Informatik, Technische Universität München,
Arcisstraße 21, D-80290 München, Germany
email: holzer@informatik.tu-muenchen.de

The number of active symbols is a measure of descriptional complexity known from the theory of ET0L systems [2,3,5] which are systems of iterated finite substitutions and, therefore, are in fact parallel string rewriting systems having a finite number of context-free production sets (tables). There, a symbol A is active (in a table P) if and only if there is a rule in P replacing A non-identically. It has been proven that, in ET0L systems, three active symbols per table are enough to generate the whole class of ET0L languages [5] whereas only one active symbol per table restricts the power of ET0L systems [2]. It is a longstanding open problem whether or not two active symbols suffice.

In this contribution, we first prove that the results for ET0L systems carry over to the special case of deterministic ET0L systems (i.e., systems of iterated homomorphisms) by a more or less substantial modification of the proof given in [5]. Then, we consider this concept of active symbols in the framework of cooperating distributed (CD) grammar systems working in t-mode of derivation, the sequential counterpart to ET0L systems, which are also seen as grammatical model of multi-agent systems [1]. Although ET0L and these CD grammar systems generate the same family of languages, we

*Part of the work was done while the author was at Département d'I.R.O., Université de Montréal, C.P. 6128, succ. Centre-Ville, Montréal (Québec), H3C 3J7 Canada. Supported by the National Sciences and Engineering Research Council (NSERC) of Canada and by the *Fonds pour la Formation de Chercheurs et l'Aide à la Recherche* (FCAR) of Québec.

show that the hierarchies with respect to the number of active symbols collapse to different levels: only one active symbol per grammar component is sufficient in case of CD grammar systems working in t-mode. This improves a result given in [4].

Furthermore, we generalize the concept of active symbols, which is rather static, to a dynamic interpretation. A symbol A is said to be dynamically measured active during a derivation if and only if A is replaced non-identically in some step of this derivation. Then, one counts the number of symbols in each table (set of productions of the system) which have to become active during some derivation yielding a terminal string. Here, only those derivations are taken into account which are the most economical ones, i.e., during which a minimal number of symbols becomes active. We prove that one dynamically measured active symbol is enough in case of both ET0L systems and CD grammar systems. Moreover, we prove that all these results carry over to the deterministic cases, as well. Finally, ET0L systems with random context and their deterministic variant are considered.

Literatur

[1] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Paun, *Grammar Systems: A Grammatical Approach to Distribution and Cooperation.* Gordon and Breach, 1994.

[2] H. C. M. Kleijn, G. Rozenberg, A Study in Parallel Rewriting Systems. *Information and Control* **44**(2), 1980, 134–163.

[3] Gh. Păun, J. Dassow, $L(ET0L_{[5]} = L(ET0L)$. *Elektronische Informationsverarbeitung und Kybernetik (Journ. Inf. Process. Cybern. EIK)* **18**(6), 1982, 345–348.

[4] S. Skalla, On the number of active nonterminals of cooperating/ distributed grammar systems. In I. Plander, ed., *Artificial Intelligence and Information-Control Systems of Robots '94. Proc. of the Sixth International Conference*, World Scientific, 1994, 367–374.

[5] T. Yokomori, D. Wood, K.-J. Lange, A Three-Restricted Normal Form Theorem for ET0L Languages. *Information Processing Letters* **14**(3), 1982, 97–100.

Sequential P-Systems

Rudolf FREUND
Institut für Computersprachen, Technische Universität Wien,
Karlsplatz 13, A-1040 Wien, Austria
email: rudi@logic.at, tel.: ++43 1 58801 18542

Abstract

We consider sequential variants of P-systems, the new model for computations using membrane structures and recently introduced by Gheorghe Păun. Using the permeabilty of the membranes for specific objects as a kind of filter turns out to be a very powerful mechanism in combination with suitable rules to be applied within the membranes. Generalized P-systems, GP-systems for short, constitute the most general model of sequential P-systems, considered in this paper. GP-systems allow for the simulation of graph controlled grammars of arbitrary type based on productions working on single objects; for example, the general results we state in this paper can immediately be applied to the graph controlled versions of context-free string grammars, n-dimensional #-context-free array grammars, and elementary graph grammars.

1 Definitions

In order to prove our results in a general setting, we use the following general notion of a grammar:

A *grammar* is a quadruple $G = (B, B_T, P, A)$, where B and B_T are sets of *objects* and *terminal objects,* respectively, with $B_T \subseteq B$, P is a finite set of *productions,* and $A \in B$ is the axiom. A production p in P in general is a partial recursive relation $\subseteq B \times B$, where we also demand that the domain of p is recursive (i.e., given $w \in B$ it is decidable if there exists some $v \in B$ with $(w, v) \in p$) and, moreover, that the range for every w is finite, i.e., for any $w \in B$, $card\,(\{v \in B \mid (w, v) \in p\}) < \infty$. The productions in P induce a derivation relation $\Longrightarrow_G$ on the objects in B. Denoting the reflexive and transitive closure of the derivation relation $\Longrightarrow_G$ by $\Longrightarrow_G^*$, the *language generated by* G is $L(G) = \{w \in B_T \mid A \Longrightarrow_G^* w\}$.

1.1 Control mechanisms

In the following, we give the necessary definitions of graph controlled grammars in our general setting. For detailed informations concerning this control mechanism as well as many other interesting results about regulated rewriting in the theory of string languages, the reader is referred to [1].

A *graph controlled grammar* is a construct $G_C = (B, B_T, (R, L_{in}, L_{fin}), A)$; B and B_T are sets of *objects* and *terminal objects*, respectively, with $B_T \subseteq B$, $A \in B$ is the axiom; R is a finite set of rules r of the form $(l(r) : p(l(r)), \sigma(l(r)), \varphi(l(r)))$, where $l(r) \in Lab(G_C)$, $Lab(G_C)$ being a set of labels associated (in a one-to-one manner) to the rules r in R, $p(l(r))$ is a production over B, $\sigma(l(r)) \subseteq Lab(G_C)$ is the *success field* of the rule r, and $\varphi(l(r))$ is the *failure field* of the rule r; $L_{in} \subseteq Lab(G_C)$ is the set of initial labels, and $L_{fin} \subseteq Lab(G_C)$ is the set of final labels. For $r = (l(r) : p(l(r)), \sigma(l(r)), \varphi(l(r)))$ and $v, w \in B$ we define $(v, l(r)) \Longrightarrow_{G_C} (w, k)$ if and only if

- **either** $p(l(r))$ is applicable to v, the result of the application of the production $p(l(r))$ to v is w, and $k \in \sigma(l(r))$,
- **or** $p(l(r))$ is not applicable to v, $w = v$, and $k \in \varphi(l(r))$.

The language generated by G_C is

$$L(G_P) = \{w \in B_T \mid (A, l_0) \Longrightarrow_{G_C} (w_1, l_1) \Longrightarrow_{G_C} \ldots (w_k, l_k),\ k \geq 1,\ w_j \in B \text{ and } l_j \in Lab(G_C) \text{ for } 0 \leq j \leq k,\ w_k = w,\ l_0 \in L_{in},\ l_k \in L_{fin}\}.$$

If the failure fields $\varphi(l(r))$ are empty for all $r \in R$, then G_C is called a *graph controlled grammar without appearance checking*. G_C is said to be of type X if the corresponding underlying grammar $G = (B, B_T, P, A)$, where $P = \{p(q) \mid q \in Lab\}$, is of type X.

2 Generalized P-systems (GP-systems)

In this section we quite informally describe the model of generalized P-systems discussed in this paper. Only the features not captured by the original model of P-systems as described in [6] and [7] will be defined in more details.

The basic part of a (G)P-system is a *membrane structure* consisting of several membranes placed within one unique surrounding membrane, the

so-called skin membrane. All the membranes can be labelled in a one-to-one manner by natural numbers; the outermost membrane (skin membrane) always is labelled by 0. In that way, a membrane structure can uniquely be described by a string of correctly matching parentheses, where each pair corresponds to a membrane. A membrane structure graphically can be represented by a Venn diagram, where two sets can either be disjoint or one set be the subset of the other one. In this representation, every membrane encloses a *region* possibly containing other membranes; the part delimited by the membrane labelled by k and its inner membranes is called *compartiment* k in the following. The space outside the skin membrane is called *outer region.*

Informally, in [6] and [7] *P-systems* were defined as membrane structures containing multisets of objects in the compartiments k as well as evolution rules for the objects. A priority relation on the evolution rules guarded the application of the evolution rules to the objects, which had to be affected in parallel (if possible according to the priority relation). The output was obtained in a designated compartiment from a halting configuration (i.e., a configuration of the system where no rules can be applied any more).

A *generalized P-system (GP-system) of type* X is a construct G_P of the following form:

$$G_P = (B, B_T, P, A, \mu, I, O, R, f)$$

where

- (B, B_T, P, A) is a grammar of type X;
- μ is a membrane structure (with the membranes labelled by natural numbers $0, ..., n$);
- $I = (I_0, ..., I_n)$, where I_k is the initial contents of compartiment k containing a finite multiset of objects from B as well as a finite multiset of operators from O and of rules from R; we shall assume $A \in I_0$ in the following;
- O is a finite set of operators (which will be described in detail below);
- R is a finite set of (evolution) rules of the form $(op_1, ..., op_k; op'_1, ..., op'_m)$ with $k \geq 1$ and $m \geq 0$, where op_1, ..., op_k, op'_1, ..., op'_m are operators from O;
- $f \in \{1, ..., n\}$ is the label of the final compartiment; we shall always assume $I_f = \emptyset$ and $R_f = \emptyset$.

The main power of GP-systems lies in the operators, which can be of the following types:

- $P \subseteq O$, i.e., the productions working on the objects from B are operators;
- $O_0 \subseteq O$, where O_0 is a finite set of special symbols, which are called *ground operators*;
- $Tr_{in} \subseteq O$, where Tr_{in} is a finite set of transfer operators on objects from B of the form $(\tau_{in,k}, E, F)$, $1 \leq k \leq n$, $E \subseteq P$, $F \subseteq P$; the operator $(\tau_{in,k}, E, F)$ transfers an object w from B being in compartiment m into compartiment k provided

  1. region m contains membrane k,
  2. every production from E could be applied to w (hence, E is also called the permitting transfer condition),
  3. no production from F can be applied to w (hence, F is also called the forbidding transfer condition);

- $Tr_{out} \subseteq O$, where Tr_{out} is a finite set of transfer operators on objects from B of the form (τ_{out}, E, F), $1 \leq k \leq n$, $E \subseteq P$, $F \subseteq P$; the operator (τ_{out}, E, F) transfers an object w from B being in compartiment m into compartiment k provided

  1. region k contains membrane m,
  2. every production from E could be applied to w,
  3. no production from F can be applied to w;

- $Tr'_{in} \subseteq O$, where Tr'_{in} is a finite set of transfer operators working on operators from P, O_0, Tr_{in}, and Tr_{out} or even on rules from R; a transfer operator $\tau_{in,k}$ moves such an element in compartiment m into compartiment k provided region m contains membrane k;
- $Tr'_{out} \subseteq O$, where Tr'_{out} is a finite set of transfer operators working on operators from P, O_0, Tr_{in}, and Tr_{out} or even on rules from R; a transfer operator τ_{out} transfers such an element in compartiment m into the surrounding compartiment.

In sum, O is the disjoint union of P, O_0, and Tr, where Tr itself is the (disjoint) union of the sets of transfer operators Tr_{in}, Tr_{out}, Tr'_{in}, and Tr'_{out}. In the following we shall assume that the transfer operators in Tr'_{in}, and Tr'_{out} do not work on rules from R; hence, the distribution of the evolution rules is static and given by I. If in all transfer operators the permitting and the forbidding sets are empty, then G_P is called a GP-system without transfer checking.

A *computation* in G_P starts with the initial configuration with I_k being the contents of compartiment k. A transition from one configuration to another one is performed by evaluating one evolution rule $(op_1, ..., op_k; op'_1, ..., op'_m)$ in some compartiment k, which means that the operators op_1, ..., op_k, are applied to suitable elements in compartiment k in the multiset sense (i.e., they are "consumed" by the usage of the rule; observe that ground operators have no arguments and are simply consumed in that way); thus we may obtain a new object by the application of a production and/or we may move elements out or into inner compartiments by the corresponding transfer operators; yet we also obtain the operators op'_1, ..., op'_m (in the multiset sense) in compartiment k.

The language generated by G_P is the set of all terminal objects $w \in B_T$ obtained in the terminal compartiment f by some computation in G_P.

3 P-systems and graph controlled grammars

The main result of this section is the simulation of graph controlled grammars of arbitrary type by GP-systems of the same type. The following result covers the case where also a kind of "context-sensitive" rules is taken into account.

Theorem 1 *Any graph controlled grammar of arbitrary type (without ac) can be simulated by a GP-system (without transfer checking) of the same type with the simple membrane structure* $[_0[_1]_1[_2]_2]_0$.

When we only allow "context-free" rules of the form $(op_1; op_2, ..., op_k)$ for some $k \geq 1$, then we need a seperate compartiment for every node of the control graph:

Theorem 2 *Any graph controlled grammar of arbitrary type (without ac) with the control graph containing n nodes can be simulated by a GP-system (without transfer checking) of the same type with the membrane structure* $[_0[_1]_1 ... [_n]_n[_{n+1}]_{n+1}]_0$ *and rules of the forms* $(op_1;)$, $(op_1; op_2)$, *and* $(op_1; op_2, op_3)$.

The general results proved above immediately apply for the string case, but as well for the objects being d-dimensional arrays and (directed) graphs:

3.1 String languages

As shown in [1], context-free graph controlled string grammars can generate any recursively enumerable string language. The theorems proved above show that GP-systems using these context-free string productions can generate any recursively enumerable string language, too.

3.2 Array languages

As was shown in [4], any recursively enumerable two-dimensional array language can even be generated by a graph controlled #-context-free two-dimensional array grammar without ac. Hence, any recursively enumerable two-dimensional array language can even be generated by a GP-system without transfer checking using #-context-free two-dimensional array productions. The same result holds true for dimensions 1 and 3, too. For $d \geq 4$, we only know that graph controlled #-context-free d-dimensional array grammars can generate any recursively enumerable d-dimensional array language; hence, recursively enumerable d-dimensional array languages can at least be generated by specific GP-systems using #-context-free d-dimensional array productions.

3.3 Graph languages

As shown in [3], any recursively enumerable graph language can be generated by graph controlled graph grammars using only the elementary graph productions of adding a new node with label K, changing the label of a node labelled by K to L, deleting a node with label K, adding a new edge labelled by a between two nodes labelled by K and M, and deleting an edge labelled by a between two nodes labelled by K and M, respectively. Hence, GP-systems using these elementary graph productions can generate any recursively enumerable graph language, too.

Acknowledgements

I gratefully acknowledge all the fruitful and interesting discussions with Gheorghe Păun on P-systems.

References

[1] J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory* (Springer, Berlin, 1989).

[2] J. Dassow and Gh. Păun, On the power of membrane computing, *Journal of Universal Computer Science* **5**, 2 (1999), pp. 33–49 (http://www.iicm.edu/jucs).

[3] R. Freund and B. Haberstroh, *Attributed Elementary Programmed Graph Grammars,* Proceedings 17th International Workshop on Graph-Theoretic Concepts in Computer Science (*LNCS* 570, Springer-Verlag, 1991), pp. 75–84.

[4] R. Freund, *Control mechanisms on #-context-free array grammars,* Mathematical Aspects of Natural and Formal Languages (Gh. Păun, ed.), World Scientific, Singapore (1994), pp. 97–137.

[5] R. Freund, Generalized P-systems, *Fundamentals of Computation Theory, FCT'99,* Iasi, 1999, (G. Ciobanu, Gh. Păun, eds.), *LNCS* 1684, Springer-Verlag (1999), pp. 281–292.

[6] Gh. Păun, Computing with Membranes, *Journal of Computer and System Sciences* **61** (2000).

[7] Gh. Păun, *Computing with Membranes: An Introduction,* Bulletin EATCS **67** (Febr. 1999), pp. 139-152.

[8] Gh. Păun, G. Rozenberg, and A. Salomaa, *DNA Computing: New Computing paradigms* (Springer-Verlag, Berlin, 1998).

[9] Gh. Păun, G. Rozenberg, and A. Salomaa, Membrane computing with external output, *Fundamenta Informaticae* **41**, 3 (2000), pp. 259–266.

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 im-

plementation 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

TANTRIX™ Rotation Puzzles are Intractable

Markus Holzer*

Institut für Informatik, Technische Universität München,
Arcisstraße 21, D-80290 München, Germany
email: holzer@informatik.tu-muenchen.de

Waltraud Holzer

Siebenbürgener Str. 3,
D-82024 Taufkirchen, Germany

TANTRIX™ is a Domino-like strategy and puzzle game made from hexagonal tiles, with painted links in combinations of red, green, blue, and yellow, and was invented in 1991 by Mike McManaway from Nelson, New Zealand. The object of the strategy game is to produce lines and loops of a specific colour. TANTRIX™ has proven immensely popular world-wide with sales exceeding two million units. During the years it won numerous international awards throughout Europe, Japan, and North America. More information on the history and how to play the game, even online, can be found under www.tantrix.com.

Challenging game problems, especially strategy games, have always been subject for mathematical investigations. This can be seen on the numerous columns on game-like problems in mathematical and popular science journals, as well as on the legion of books on games. Complexity considerations on classical games like Chess [5], Go [3], or Hex [4] were done in the early 1980's, but also Domino-like problems have been investigated [1]. For instance, Lewis and Papadimitriou [2] have shown that there is a Domino puzzle played with square tiles which is **NP**-complete. The only restriction in this games is, that it is not allowed to rotate a given tile to make it fit. This brings us to the subject of this paper, because TANTRIX™ is a strategy

*Supported in part by the National Sciences and Engineering Research Council (NSERC) of Canada and by the *Fonds pour la Formation de Chercheurs et l'Aide à la Recherche* (FCAR) of Québec. Part of the work was done while the author was at Département d'I.R.O., Université de Montréal, C.P. 6128, succ. Centre-Ville, Montréal (Québec), H3C 3J7 Canada.

as well as a puzzle game.

There are six different types of puzzles described in the game description of TANTRIX™, which vary in difficulty (novice to master). We consider the *rotation puzzle* and analyze its complexity. Simply speaking, in a rotation puzzle the location of the tiles is given, and the only possibility to make them fit is to rotate them. Although, the problem is very restrictive, it has a surprisingly high complexity, namely we show that it is intractable. Speaking in terms of complexity, we prove the following theorem:

Theorem 1 *The* TANTRIX™ *rotation puzzle is* **NP***-complete.*

To this end, we show how to simulate a circuit with AND- and NOT-gates *via* a TANTRIX™ rotation puzzle. This is used to construct a puzzle that has a solution if and only if there is an assignment to the input variables such that the original circuit evaluates to *true*.

References

[1] E. Grädel. Domino games and complexity. *SIAM Journal on Computing*, 19(5):787–804, 1990.

[2] H. Lewis and C. Papadimitriou. *Elements of the theory of computation.* Prentice-Hall, 1982.

[3] D. Lichtenstein and M. Sipser. GO is polynomial-space hard. *Journal of the ACM*, 27(2):393–401, April 1980.

[4] S. Reisch. Hex ist PSPACE-vollständig. *Acta Informatica*, 15:167–191, 1981.

[5] J. A. Storer. On the complexity of chess. *Journal of Computer and System Sciences*, 27(1):77–100, August 1983.

Parsen von erweiterten regulären Ausdrücken

Andreas Klein

Reguläre Ausdrücke sind ein mächtiges Werkzeug für Textmanipulation in vielen modernen Computersprachen. Allerdings ist es für viele praktische Beispiele notwendig, gewisse Erweiterungen zuzulassen.

Es stellt sich heraus, daß es sehr schwer ist, solche erweiterten regulären Ausdrücke mit deterministischen endlichen Automaten zu behandeln. Daher benutzen viele Programme Algorithmen, die auf nichtdeterministischen Automaten basieren. Dies führt jedoch im schlimmsten Fall zu exponentieller Laufzeit.

In diesem Vortrag bespreche ich die folgenden Erweiterungen und stelle einen Algorithmus vor, mit dem sich diese erweiterten regulären Ausdrücke in linearer Zeit parsen lassen.

Einfangende Klammern
Man ist häufig nicht nur an dem Teilstring interessiert, der durch den regulären Ausdruck abgedeckt wird. Oft will man auch wissen, welche Teile von den einzelnen Unterausdrücken abgedeckt werden. (Beispielsweise ist man bei dem regulären Ausdruck `"([^"]*)"` für einen String in Anführungszeichen auch an dem erkannten String ohne die Anführungszeichen interessiert, d.h. an dem Teil, der durch `([^"]*)` abgedeckt wird.)

Nicht gierige Operatoren
Normalerweise sucht man den längsten passenden Teilstring. (Dieses Verhalten von regulären Ausdrücken wird im POSIX Standart festgelegt.) Manchmal ist man jedoch am kürzesten möglichen Treffer interessiert. (Beispiel: Wir suchen nach C Kommentaren mit Hilfe von `/\*.*\*/`.) Daher definiert Perl zusätzlich zu der normalen (gierigen) Wiederholung (`*`) die nicht gierige Wiederholung (`*?`).

Ein weiterer Unterschied liegt in der Behandlung des Oder–Operators (|). Man kann entweder die längste Alternative bevorzugen (gieriges oder) oder die erste Alternative.

Look–Ahead Operatoren
Es ist häufig nützlich, in einem regulären Ausdruck auch auf die unmittelbar folgenden Zeichen Bezug nehmen zu können. Es ist relativ leicht, solche Look–Ahead Operatoren für das Ende eines Treffers zu schreiben. Der allgemeinste Fall erfordert jedoch wesentlich mehr Arbeit.

Deterministische Turingmaschinen zwischen Real- und Linearzeit

Martin Kutrib
Institut für Informatik
Universität Gießen
Arndtstr. 2, 35392 Gießen
kutrib@informatik.uni-giessen.de

Viele der Ergebnisse für deterministische Turingmaschinen mit höchstens linearer Zeitbeschränkung stammen aus den Anfangszeiten der Komplexitätstheorie. Während für fast alle nichtdeterministischen Modelle und deterministische Einband-Maschinen bekannt ist, daß die Realzeit- und Linearzeitklassen zusammenfallen, ergeben sich signifikante Unterschiede für deterministische Maschinen mit mindestens zwei Bändern: Hier sind die Realzeitklassen jeweils echt in den Linearzeitklassen enthalten.

Für den Bereich dazwischen liefern die Zeithierarchiesätze keine Antwort. Z.B. ist für k-Band-Turingmaschinen die echte Inklusion $\mathsf{DTIME}_k(t_1) \subset \mathsf{DTIME}_k(t_2)$ bekannt, sofern $k \geq 2$, $t_1 \in o(t_2)$ und t_2 zeitkonstruierbar sind. (Für Mehrband-Maschinen muß $t_1 \cdot \log(t_1) \in o(t_2)$ gelten.) Die Hierarchiebedingung $t_1 \in o(t_2)$ wird aber aufgrund der Möglichkeit zur linearen Beschleunigung zur kleinstmöglichen. Es gilt bekanntlich $\mathsf{DTIME}(t) = \mathsf{DTIME}(Lin(t))$ für Mehrband- und k-Band-Maschinen, sofern $t \geq c \cdot n$ mit $c > 1$ ist. Andererseits sind damit wegen der Bedingung $c > 1$ ebenfalls keine Aussagen über den Bereich zwischen Real- und Linearzeit möglich.

Wir betrachten die Zeitkomplexitätsklassen $\mathsf{DTIME}(n + r(n))$ mit sublinearen Abbildungen r und zeigen eine unendliche, echte Hierarchie über die r:

Für zwei Abbildungen $r : \mathbb{N}_0 \to \mathbb{N}$ und $r' : \mathbb{N}_0 \to \mathbb{N}$ mit $r' \in o(r)$ und r^{-1} zeitkonstruierbar gilt: $\mathsf{DTIME}(n + r'(n)) \subset \mathsf{DTIME}(n + r(n))$.

Da eine Beschleunigung von $n + r(n)$ nach $n + \varepsilon \cdot r(n)$, $\varepsilon > 0$, möglich ist, folgt, daß die Hierarchie dicht ist.

Die untersuchten Klassen sind weder unter Konkatenationsabschluß noch unter Konkatenation abgeschlossen. Beide Abschlußeigenschaften stimmen mit denen der Realzeit-Maschinen überein und sind für Linearzeit offen.

Die Ergebnisse lassen sich axiomatisch auf andere Berechnungsmodelle übertragen. Sie gelten z.B. auch für mehrdimensionale Turingmaschinen und zum Teil für iterative Arrays.

Auf Zellularautomaten basierende Bilderzeugung und -kompression

Jan-Thomas Löwe
Institute für Informatik
Universität Giessen
Arndtstr. 2, D-35392 Giessen
email: Jan-Thomas.Loewe@informatik.uni-giessen.de

In Computersystem werden Bilder normalerweise durch eine Sequenz von Helligkeitswerten für jede Farbe und jedes Pixel beschrieben. Der Nachteil dieser Beschreibungsform besteht im hohen Speicherbedarf, der für jedes Bild benötigt wird. Daher ist es von Interesse Methoden anzuwenden und zu entwickeln, die die Informationsmenge auf das Wesentliche reduzieren, d.h. das Bild komprimieren.

Es gibt mehrere universelle Kompressions-Algorithmen, die auf einen Datenstrom angewendet werden können, z.B. die LZ-Kompression in [3]. Doch diese berücksichtigen nicht einige besondere Eigenschaften und Charakteristiken von Bildern.

In [1, 2] benutzen Culik und Kari einen Kompressionsalgorithmus, der auf (gewichteten) endlichen Automaten basiert. Der endliche Automaten beschreibt dabei das ursprüngliche Bild, indem Bilder als endliche Sprache aufgefaßt werden. Kompression bedeutet, einen Automaten zu konstruieren, der ein Eingabewort erhält, das der Postion eines Pixels entspricht, und eine Ausgabe produziert (einen Graustufenwert).

Zellularautomaten sind in Bezug auf Sprachverarbeitung leistungsfähiger als endliche Automaten. Aus diesem Grund sind wir daran interessiert herauszufinden, wie (eventuell auch modifizierte) Zellularautomaten zur Bilderzeugung und -kompression benutzt werden können.

Dazu betrachten wir zwei-dimensionale Zellularautomaten. Jede Zelle entspricht einem Pixel des Bildes. Das Ziel ist die Verwendung von Signalen, um eine Zelle bzw. ein Pixel einzufärben. Signale erlauben es uns Teilbilder zu kombinieren, skalieren, rotieren um daraus neue Teilbilder zu generieren. Daraus erhoffen wir uns Speicher für Teilbilder einzusparen, die aus schon gespeicherten generiert werden können.

Literatur

[1] K. Culik II, J. Kari *Image Compression Using Weighted Finite Automata.* Comput. Graphics 17 (1993), 305-313.

[2] K. Culik II, J. Kari *Inference Algorithms for WFA and Image Compression.* In Fractal Image Compression, ed. Y. Fisher, Springer, 1994, 245-262.

[3] J. Ziv, A. Lempel *A Universal Algorithms for Sequential Data Compression.* IEEE Transactions on Information Theory Vol. IT-23 No. 3 (1977) 337-343.

A Remark on the Succinctness of Descriptions of Context-Free Languages by Cooperating Distributed Grammar Systems

Bernd Reichel
Fakultät für Informatik
Otto-von-Guericke-Universität
Postfach 4120
D-39016 Magdeburg
Germany
e-mail: reichel@iws.cs.uni-magdeburg.de

Cooperating distributed grammar systems (CD grammar systems for short) can be considered as a generalization of context-free grammars, where the set of rules is splitten in a number of parts. Then for the derivation process there are several control mechanisms, so-called derivation modes. CD grammar systems are introduced in [1] and have for some derivation modes a larger generative power than context-free grammars.

Considering only the family of context-free languages the question arises if CD grammar systems have any advantages over context-free grammars in respect of descriptions. This question was investigated in [2] concerning the descriptional measures number of variables (Var), number of productions (Prod) and number of symbols (Symb) for several derivation modes.

The case of most increase of effiency is reached, if there is a sequence of languages such that its descriptional measures with respect to CD grammar systems are bounded by a constant and its descriptional measures with respect to context-free grammars are unbounded. For the derivation mode t (i.e. each component of rules performs derivations on the sentential form as long as possible) this best increase has been achieved in [2] for the complexity measure Var, whereby for the complexity measures Prod and Symb indeed an increase has been shown but it has been left open whether or not the best increase is reachable.

In this talk we will answer these open questions concerning the complex-

ity measures Prod and Symb with respect to the derivation mode t, i.e, we show that we can reach this best increase for the complexity measure Prod and show that we can not reach this best increase for the complexity measure Symb.

Moreover, we will tighten the result for the complexity measure Var by allowing only CD grammar systems with a bounded number of components. Also, for the complexity measures Prod and Symb we reached the result by allowing only CD grammar systems with a bounded number of components.

Moreover, in all cases we use only a bounded alphabet (unary for Prod and Symb, binary for Var).

Acknowlegdement. I thank Henning Bordihn for turning my attention to this subject.

References

[1] E. Csuhaj-Varjú and J. Dassow. On cooperating/distributed grammar systems. *Journal of Information Processing and Cybernetics (EIK)* **26** (1990) 1/2, 49–63.

[2] J. Dassow and Gh. Păun. On the succinctness of descriptions of context-free languages by cooperating/distributed grammar systems. *Computers and Artificial Intelligence* **10** (1991) 6, 513–527.

Die $\#a = \#b$ Bilder sind erkennbar *

Klaus Reinhardt
Wilhelm-Schickhard Institut für Informatik, Universität Tübingen
Sand 13, D-72076 Tübingen, Germany
e-mail: reinhard@informatik.uni-tuebingen.de

Zusammenfassung

Giammarresi, Restivo, Seibert und Thomas definieren eine Bildsprache als lokal, wenn sie beschrieben werden kann als Menge der Bilder, bei denen jedes 2x2 Unterbild zu einer festgelegten Menge von 2x2 Kacheln gehört. Die Menge der erkennbaren Sprachen ist der Abschluss der lokalen Sprachen unter Projektion.

Es wird gezeigt, dass die Sprache der Bilder aus gleich vielen a's wie b's erkennbar ist. (Bei nicht entartetem Höhen-Breiten Verhältnis) Das bedeuted dass das Zählen in rechteckigen Gittern in existentieller monadischer Logik zweiter Stufe definierbar ist.

1 Einleitung und Definition

In [GRST94] werden *Bilder* als 2-dimensionale Wörter über einem Alphabet Σ definiert. Die Menge der Bilder der Größe (m, n) wird bezeichnet durch $\Sigma^{m,n}$. Eine *Bildsprache* ist eine Teilmenge $\Sigma^{*,*} := \bigcup_{m,n\geq 0} \Sigma^{m,n}$. Für ein $p \in \Sigma^{m,n}$, entsteht $\hat{p} \in \Sigma^{m+2,n+2}$ durch Hinzufügen eines Rahmens aus $\# \notin \Sigma$. Eine Bildsprache $L \subseteq \Gamma^{*,*}$ heisst *lokal*, wenn es ein Δ gibt mit $L = \{p \in \Gamma^{*,*} | T_{2,2}(\hat{p}) \subset \Delta\}$. Eine Bildsprache $L \subseteq \Gamma^{*,*}$ heisst *hv-lokal*, wenn es ein Δ gibt mit $L = \{p \in \Gamma^{*,*} | T_{1,2}(\hat{p}) \cup T_{2,1}(\hat{p}) \subset \Delta\}$. Eine Bildsprache $L \subseteq \Sigma^{*,*}$ heisst *erkennbar*, wenn es eine Projektion $\pi : \Gamma \to \Sigma$ und eine lokale Bildsprache $L' \subset \Gamma^{*,*}$ mit $L = \pi(L')$ gibt.

Nach [LS97b] erhält man die erkennbaren Sprache ebenso, wenn man anstelle einer lokalen eine hv-lokale Urbildsprache verwendet. Eine Bildsprache ist ganau dann erkennbar [GRST94], wenn sie in existentieller monadischer Logik zweiter Stufe definierbar ist. Betrachtet man Wörter am Bildrand

*Mehr unter http://www-fs.informatik.uni-tuebingen.de/~reinhard/recpigl.ps

bzw. an der Kante eines Hyperwürfels, so erhält man gemäß [LS97a] bzw. [Bor99] die kontextsensitiven Sprachen bzw. NP.

Im eindimensionalen Fall ist Zählen ein Prototypkonzept für Nichterkennbarkeit ($\equiv$ Nichtregularität); durch das Hinzufügen einer zusätzlichen Dimension wird Zählen jedoch möglich, für Bilder wurde aber bisher vermutet, dass eine dritte Dimension nötig sei um Zählen zu können. Die nichtuniforme Zählmethode in [Rei98] konnte nur erklären, warum Versuche folgendes Theorem zu wiederlegen, fehlschlugen.

Erlaubt man Bilder der Größe (m,n), die nicht die Restriktion $m \leq f(n)$ und $n \leq f(m)$ für eine Funktion $f \in 2^{O(n)})$ erfüllen, so sind mehr als exponentiell viele Anzahldifferenzen in einer Bildhälfte möglich und aus einem Lemma in [Mat98], welches besagt, dass in erkennbaren Bildsprachen höchstens exponentiell viel Information von einer Bildhälte zur andern fliesen kann, folgt:

Korollar *Die Sprache der Bilder über $\{a,b\}$, mit gleicher Anzahl von a's wie b's ist nicht erkennbar.*

Hauptergebnis dieser Arbeit ist das folgende Theorem:

Definition Die Bildsprache $L_=$ (bzw. $L^c_=$) ist die Menge der Bilder über $\{a,b\}$ (bzw. $\{a,b,c\}$) mit gleicher Anzahl von a's wie b's und einer Größe (n,m) mit $m \leq 2^n$ und $n \leq 2^m$.

Bem: Genauso könnte auch eine größere Konstante k anstelle von 2 als Basis verwendet werden, d.h. es gibt keine Lücke zum Korollar.

Theorem *Die Bildsprachen $L_=$ und $L^c_=$ sind erkennbar.*

Wesentliche Idee des Beweises ist die Konstruktion eines 'zählenden Flusses' im Urbild, der lokal nur konstante Kapazität besitzt aber durch unterschiedliche Wertigkeiten in der Weise eines Binärzählers große Zahlen aufnehmen kann. Um überall im Bild auch einzelne Werte aufnehmen zu können, müssen ähnlich dem Zähler in [Für82] niederwertige Flusswege häufiger auftreten wie höherwertige. Schwierigkeit dabei ist, lokal die Positionen zu erkennen, wo ein Fluss mit z.B. Wertigkeit 4^i einem Fluss mit Wertigkeit 4^{i+1} begegnet um die Möglichkeit eines Übertrages zu schaffen. Zu diesem Zweck wird zunächst ein 'Skelett' konstruiert.

2 Reduktion auf ungerade Positionen und Zweierpotenzquadrate

Wir betrachen eine Abbildung $e : \Sigma \mapsto \Sigma_e^{i,j}$ als erweitert um ein Bild der Größe (m,n) über Σ auf ein Bild der Größe (im,jn) über Σ_e abzubilden.

Lemma *Eine Bildsprache L über Σ ist erkennbar, wenn $e(L)$ erkennbar und e injectiv ist.*

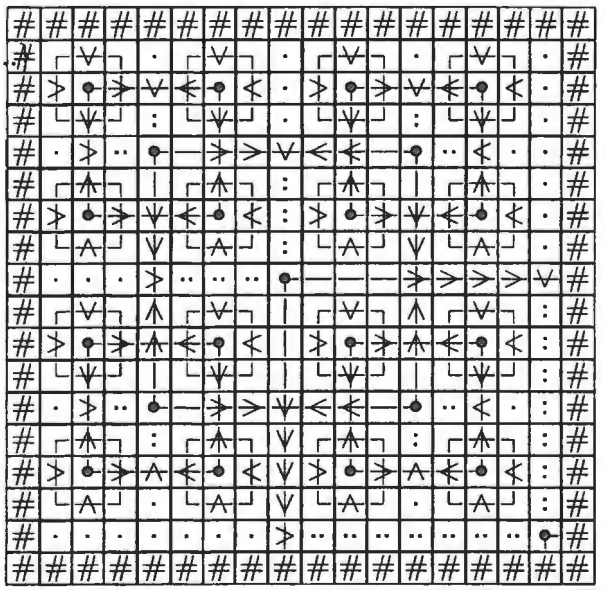
Um zu zeigen dass $L_{=}^{c}$ und somit $L_{=} = L_{=}^{c} \cap \{a,b\}^{*,*}$ erkennbar ist, genügt es also für $e : \{a,b,c\} \mapsto \{a,b,c,d\}^{2,2}$ mit $e(x) = \begin{array}{|c|c|}\hline x & d \\ \hline d & d \\ \hline \end{array}$ für $x \in \{a,b,c\}$ zu zeigen dass $e(L_{=}^{c})$ erkennbar ist. D.h, man braucht sich nur um Bilder in $\{\begin{array}{|c|c|}\hline x & d \\ \hline d & d \\ \hline \end{array} \mid x \in \{a,b,c\}\}^{*,*}$ zu kümmern.

Für ein Bild der Größe (m,n) mit o.B.d.A. $m > n$ wähle das kleinste i mit $2^i \geq n$ und das kleinste j mit $j2^i \geq m$. Mit Hilfe einer Simulation des so gefalteten Bildes in 4 Schichten reduziert sich das Problem darauf zu zeigen dass $e(L_{=}^{c}) \cap \bigcup_{i,j} \Sigma^{j2^i,2^i}$ erkennbar ist.

3 Ein 'Skelett' für Zweierpotenzquadrate

Das *Skelett* beschreibt ein Quadrat, dessen Ecken jeweils von einem Quadrat der halben Größe umgeben sind, es wird beschriben durch eine hv-lokale Sprache L_S über dem Alphabet $\Sigma_S = \{\urcorner, \ulcorner, \llcorner, \lrcorner, \bullet, \bullet, \bullet, \bullet, \cdot, \cdot\cdot, :, \vee, \wedge, \not<, \not>, \curlyvee, \curlywedge, \not\wedge, \not\vee, |, \curlyvee, \curlywedge, -, <, >\}$.

Uns interessiert nur der Schnitt $L_S \cap L_R := \{p_1, p_2, \ldots$ mit der erkennbaren Sprache $L_R = \{\begin{array}{|c|c|}\hline x & y \\ \hline z & w \\ \hline \end{array} \mid$ $x \in \{\urcorner, \ulcorner, \llcorner, \lrcorner\}$, $y,z,w \in \{\bullet, \bullet, \bullet, \bullet, \cdot, \cdot\cdot, :, \vee, \wedge, \not<, \not>, \curlyvee, \curlywedge, \not\wedge, \not\vee, |, \curlyvee, \curlywedge, -, <, >\}\}^{*,*}$.



Ein Beispielbild ist $\hat{p}_4 =$

Wir definieren L_S durch $\Delta_S := T_{1,2}(\hat{p}_4) \cup T_{2,1}(\hat{p}_4) \cup \{$

$\}$.

Für jedes i gibt es genaue ein BIld p_i der Größe $(2^i, 2^i)$ in $L_S \cap L_R$ und für alle $i > 1$, $r, c \geq 0$ hat das Teilbild q der Größe $(2^i - 1, 2^i - 1)$ beginnend bei $p(c2^i + 1, r2^i + 1)$ bis auf 90^o-Rotation folgendes Aussehen an der Aussenseite: Die obere Reihe ist periodisch gefüllt mit $q(t, 1) =$ ⌜ (bzw. ⊤, ¬, ·) wenn $tMod4 = 1$ (bzw. 2,3,0) und $t < 2^i$. Gleiches gilt in 90^o-rotationssymmetrischer Weise mit der Ausnahme $q(2^{i-1}, 2^i - 1) =$ ⊥ und $q(2^i - 1, 2^{i-1}) =$ → wenn $i > 2$ und $q(2^{i-1}, 2^i - 1) =$ ↑ und $q(2^i - 1, 2^{i-1}) =$ ↠ wenn $i = 2$.
Auf http://www-fs.informatik.uni-tuebingen.de/~reinhard/picgen.html befindet sich ein Java-Program, welches demonstriert, dass ein Symbol meist bereits durch seinen linken und oberen Nachbarn erzwungen wird, in den übrigen Fällen führt eine falsche Auswahl früher oder später zum Abbruch.

4 Der zählende Fluss für Zweierpotenzquadrate

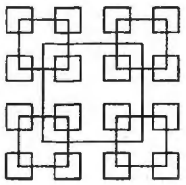
Lemma $e(L_=^c) \cap \bigcup_i \Sigma^{2^i,2^i}$ *ist erkennbar.*
Beweis: Wir definieren die Sprache L_F so dass $e(L_=^c) \cap \bigcup_i \Sigma^{2^i,2^i} = L_F \cap e(\{a, b, c, \}^{*,*})$ gilt: Die Kapazität von einer Zelle zu seinem Nachbarn erlaubt einen Fluss von -9 bis 9: $\Sigma_F := \Sigma_S \times \{-9, -8, ..., 9\}^4$. Ausserdem erlauben wir nur Symbole $(x, l, r, u, d) \in \Sigma$ mit:
$\pi(x, l, r, u, d) := a$ wenn $x \in \{\neg\ ,\ \ulcorner,\ \llcorner, \lrcorner\} \wedge l + r + u + d = 1$,
$\pi(x, l, r, u, d) := b$ wenn $x \in \{\neg\ ,\ \ulcorner,\ \llcorner, \lrcorner\} \wedge l + r + u + d = -1$,
$\pi(x, l, r, u, d) := c$ wenn $x \in \{\neg\ ,\ \ulcorner,\ \llcorner, \lrcorner\} \wedge l + r + u + d = 0$,
$\pi(x, l, r, u, d) := d$ wenn $x \in \{$ •, •, •, • $\} \wedge l + r + u + d = 0$,
oder $x \in \{\cdot, \cdot\cdot, :, |, \forall, \uparrow, —, \leftarrow, \rightarrow\} \wedge l = -r \wedge u = -d$,
oder $x \in \{\top, \bot, \downarrow, \uparrow\} \wedge l + r + 4u + 4d = 0$,
oder $x \in \{\not<, \not>, \not\leftarrow, \not\rightarrow\} \wedge 4l + 4r + u + d = 0$. Die Kachelmenge

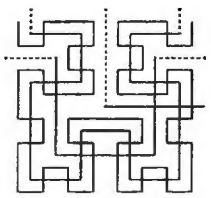
$$\Delta_F := \{\boxed{\begin{matrix} f \\ g \end{matrix}} \mid \boxed{\begin{matrix} f_1 \\ g_1 \end{matrix}} \in \Delta_S, f = (f_1, l, r, u, d), g = (g_1, l', r', d, d')\}$$
$$\cup \{\boxed{f\ g} \mid \boxed{f_1 g_1} \in \Delta_S, f = (f_1, l, r, u, d), g = (g_1, r, r', u', d')\}.$$

sorgt für die korrekte Fortsetzung des Flusses. Die Symbole ¬ , ⌜, ⌞, ⌟ erlauben Quellen und Senken des Flusses; sie kommen nur in ungeraden Zeilen und Spalten vor und haben die Wertigkeit 1; •, •, •, • kommen nur vor, wo Zeile und Spalte die gleiche Wertigkeit haben; ·, ··, :, |, ∀, ↑, —, ←, → kommen nur vor, wo Zeile und Spalte völlig verschiedene Wertigkeiten haben und ⊤, ⊥, ↓, ↑, ≮, ≯, ↚, ↛ kommen nur vor, wo Quadrate des Skelettes oder deren Verlängerung ein Quadrat der halben Größe schneidet; hier

kann ein Übertrag im Verhältnis 4 zu 1 stattfinden. Im Allgemeinen hat für jedes j und i die $2^{i-1} + j \cdot 2^i$ -te Zeile bzw. Spalte dei Wertigkeit 4^i.

Der Fluss in einem Bild $p \in L_F \cap e(\{a, b, c, \}^{*,*})$ beweist somit $p \in e(L^c_=)$. Umgekehrt kann in einem Bild $p \in e(L^c_=) \cap \bigcup_i \Sigma^{2^i,2^i}$ ein Fluss entlang Iterationen der Hilbertkurve konstruiert werden:





Eine Kurve, die die Wertigkeit 4^i representiert, kann Überträge von der Kurve mit Wertigkeit 4^{i-1} erhalten, wird aber immer rechtzeitig vor Überschreiten ihrer Kapazität der Kurve mit Wertigkeit 4^{i-1} begegnen.

Mit Hilfe einer weiteren, einfacheren Zählmethode, können die aus Quadraten herauskommenden Flüsse in das nächste weitergeleitet werden, wodurch auch $e(L^c_=) \cap \bigcup_{i,j} \Sigma^{j \cdot 2^i,2^i}$ erkennbar wird.

Literatur

[Bor99] B. Borchert. A formal languages characterization of NP. Manuscript at http://math.uni-heidelberg.de/logic/bb/papers/NP-char.ps, 1999.

[Für82] Martin Fürer. The tight deterministic time hierarchy. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 8–16, San Francisco, California, 5–7 May 1982.

[GRST94] Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. Monadic second-order logic over pictures and recognizability by tiling systems. In *Proceedings of the 11th STACS 94 (Caen, France, February 1994)*, LNCS 775, pages 365–375, 1994. Springer-Verlag.

[LS97a] M. Latteux and D. Simplot. Context-sensitive string languages and recognizable picture languages. *Information and Computation*, 138(2):160–169, 1 November 1997.

[LS97b] M. Latteux and D. Simplot. Recognizable picture languages and domino tiling. *Theoretical Computer Science*, 178(1-2):275–283, 1997. Note.

[Mat98] Oliver Matz. On piecewise testable, starfree, and recognizable picture languages. In Maurice Nivat, editor, *Foundations of Software Science and Computation Structures*, volume 1378 of *Lecture Notes in Computer Science*, pages 203–210. Springer, 1998.

[Rei98] K. Reinhardt. On some recognizable picture-languages. In L. Brim, editor, *Proceedings of the 23th Conference on Mathematical Foundations of Computer Science*, number 1450 in Lecture Notes in Computer Science, pages 760–770. Springer-Verlag, August 1998.

Node Replacement D0L Systems

Ralf Stiebe
Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik
Kurt-Mothes-Straße 1
D-06120 Halle(Saale)
email: stiebe@informatik.uni-halle.de

1 Introduction

Node replacement is one of the basic approaches to the rewriting of graphs. For an overview on sequential node replacement, see the handbook article of ENGELFRIET/ROZENBERG [3]. Parallel variants of node replacement grammars were discussed in papers of JANSSENS/ROZENBERG/VERRAEDT [4, 5], and in the dissertation of RÖDER [7]. In this paper we discuss those node replacement grammars that are generalizations of D0L and HD0L systems.

The extension of (H)D0L systems to graphs is naturally obtained by generalizing the notion of a homomorphism to labeled graphs. Such a *generalized homomorphism* consists of a *node replacement*, substituting nodes by (disjoint) daughter graphs, and a set of *connection instructions*, substituting an edge by a set of edges between the respective daughter graphs. A node replacement D0L systems (NR-D0L systems) is defined by an axiom graph, and a generalized homomorphism, which is iterated to obtain the graph language. In analogy to classic L systems, we can define NR-HD0L systems where the graphs generated by a NR-D0L system are subject to a final homomorphism. The respective families of graph languages generated are denoted by $\mathcal{G}$(NR-D0L), $\mathcal{G}$(NR-HD0L).

2 Generative Power

Considering the generative power of NR-HD0L systems, an important result follows from the simple fact that the sequences of numbers of nodes are HD0L growth sequences (N-rational sequences).

Theorem 1 *For any NR-HD0L system, there is a constant k such that the number of different graphs with the same number of nodes is bounded by k.*

This generalizes the fact that HD0L languages are *slender* [6].

Surprisingly, NR-(H)D0L systems are equivalent to the seemingly completely different approach of graph generation by edge grammars, due to BERMAN [1]. An edge grammar is a grammar that generates a binary word relation, or equivalently, a language over some alphabet $X \times X$. A pair (v, w) with $|v| = |w| = n$ is interpreted as an edge in the n-th graph. The motivation for the introduction of edge grammars is that important graph families (hypercubes, shuffle-exchange graphs, ...) can easily be described by such word relations. The families of graph languages generated by [prefix-closed] regular languages over alphabets of the form $X \times X$ are denoted by $\mathcal{G}$([Pref-]REG). To establish the claimed equivalence we consider the unlabeled versions of NR-(H)D0L graph languages.

Theorem 2 $\mathcal{G}(NR\text{-}D0L) = \mathcal{G}(Pref\text{-}REG)$,
$\mathcal{G}(NR\text{-}HD0L) = \mathcal{G}(REG)$.

Closure and Decidability Results

Given a graph generating device, it is of interest to decide whether the obtained graph language contains graph with particular properties. A related question is whether a family of graph language is closed under a graph-theoretic property. We are able to show many results for NR-HD0L systems, using the representation by regular languages over alphabets of pairs. The proofs can be found in the author's dissertation [8].

Theorem 3 *For a NR-HD0L system Γ and a graph property φ expressible in first order logic*[1]*, the set $\{n : G_n(\Gamma) \models \varphi\}$ is effectively ultimately periodic.*

Corollary 4 *For a NR-HD0L system Γ and a graph property φ expressible in first order logic,*

1. *one can construct a NR-HD0L system Θ generating exactly the graphs derived by Γ with property φ.*
2. *it is decidable whether some graph/all graphs in $\boldsymbol{G}(\Gamma)$ has/have property φ.*

[1] For the expression of graph properties by logic, see the article of COURCELLE [2].

Theorem 5 *For a NR-HD0L system, it is decidable whether the maximal degree of the generated graphs is bounded.*

Theorem 6 *1. $\mathcal{G}$(NR-HD0L) is not closed under the property of being acyclic (connected, k-colorable, Eulerian, ...).*

2. For NR-HD0L systems, it is undecidable whether some graph/all graphs has/have a property mentioned above.

3. The equivalence problem is undecidable for NR-HD0L systems.

It is an open question which of the negative results in Theorem 6 are valid for NR-D0L system. As a first step, we can give results for the property of being acyclic.

Theorem 7 *For a NR-D0L system Γ,*

1. the set $\{n : G_n(\Gamma) \models \varphi\}$ is ultimately periodic (but not effectively).

2. it is decidable whether some graph generated by Γ is acyclic.

3. it is undecidable whether all graphs generated by Γ are acyclic.

3 Conclusions

We have established an equivalence between NR-(H)D0L systems and edge grammars. Many theoretical results could be shown using the representation by edge grammars, while NR-D0L systems seem to be more convenient for the specification of graph languages.

References

[1] Francine Berman. Edge grammars and parallel computation. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, pages 214–223, 1983.

[2] Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, pages 313–400. World Scientific, Singapore, 1997.

[3] Joost Engelfriet and Grzegorz Rozenberg. Node replacement graph grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, pages 1–94. World Scientific, Singapore, 1997.

[4] Dirk Janssens, Grzegorz Rozenberg, and R. Verraedt. On sequential and parallel node-rewriting graph grammars I. *Computer Graphics and Image Processing*, 18:297–304, 1982.

[5] Dirk Janssens, Grzegorz Rozenberg, and R. Verraedt. On sequential and parallel node-rewriting graph grammars II. *Computer Graphics and Image Processing*, 18:279–304, 1982.

[6] T. Y. Nishida and A. Salomaa. Slender 0L languages. *Theoretical Computer Science*, 158(1-2):161–176, May 1996.

[7] Hans-Joachim Röder. *Parallele BNLC-Graphgrammatiken.* Dissertation, Universität Passau, 1992.

[8] Ralf Stiebe. *Untersuchungen zu Kantengrammatiken und Valenzgrammatiken.* Dissertation, Martin-Luther-Universität Halle-Wittenberg, 2000.

Fixpunkte von Morphismen und Normalformen von Ersetzungssystemem

Johannes Waldmann
Institut für Informatik, Universität Leipzig
Augustusplatz 10-11, D-04109 Leipzig, Germany
joe@informatik.uni-leipzig.de
http://www.informatik.uni-leipzig.de/~joe/

Zusammenfassung

Wir können Bäume, die sich als Normalformen von Ersetzungssystemen ergeben, in einigen Fällen auch durch iterierte Morphismen beschreiben, und unendliche Normalformen von top–terminierenden Systemen durch Fixpunkte von Morphismen.

1 Morphismen und HD0L

Das bekannte Fibonacci–Wort $f = 01001010010010\ldots$ (siehe [Lot97b]) ist der Fixpunkt des Morphismus ϕ:

$$f = \lim f_n,\ f_n = \phi^n(0),\ \phi : 0 \mapsto 01, 1 \mapsto 0$$

Das folgende Haskell–Programm berechnet die Liste der Buchstaben von f:

```
phi :: [ Char ] -> [ Char ]
phi ('0' : rest) = '0' : '1' : phi rest
phi ('1' : rest) = '0' : phi rest

f :: [ Char ]
f = '0' : t where t = '1' : phi t
```

Das können wir ganz leicht als eine Wort–Ersetzungs–System R auffassen mit der Regelmenge

$$R : T \to 1\Phi T, \Phi 0 \to 01\Phi, \Phi 1 \to 0\Phi$$

Dieses System terminiert nicht, aber bei fairer Reduktion (kein Redex lebt unendlich lange) wandern die Redexe nach rechts, und wir erhalten im Limes das Fibonacci–Wort $f = R^\infty(0T)$.
Die Konstruktion war generisch; offensichtlich können wir so jeden Fixpunkt eines monotonen Morphismus als Links Normalform eines Ersetzungssystems darstellen.
Umgekehrt finden wir in einigen Fällen zu einem gegebenem Wort einen passenden Morphismus. Betrachten wir

$$q = 1\ 1\ 0\ 1\ 00\ 1\ 000\ 1\ldots$$

aus immer längeren Blöcken von 0.
Dieses können wir durch einen iterierten Morphismus f erzeugen,

$$f : S \mapsto S1, 1 \mapsto 10, 0 \mapsto 0$$

von dem wir anschließend ein homomorphes Bild nehmen (um das „Startsymbol“ S zu löschen)

$$g : S \mapsto \epsilon, 0 \mapsto 0, 1 \mapsto 1; \quad q = \lim g(f^n(S))$$

Das Wachstum dieses Morphismus ist polynomiell (quadratisch) beschränkt.
Das Tupel (g, f, S) ist ein HD0L System. (Für Definitionen fehlt hier der Platz, siehe jedoch [KRS97] .)

Ein weiteres Beispiel (mit exponentiellem Wachstum) benötigen wir gleich:

$$N_k = h^k(R) \textbf{ wobei } h : R \mapsto RL, L \mapsto RR$$

Beispiel: $N_2 = RLRR, N_3 = RLRRRLRL$. Offensichtlich ist $|N_k| = 2^k$.
Wir bezeichnen mit w' das Wort w ohne seinen letzten Buchstaben. Dann zeigt man leicht durch Induktion

$$N'_{k+1} = \textbf{wenn } 2 \mid k \textbf{ dann } N'_k R N'_k \textbf{ sonst } N'_k L N'_k$$

Im Beispiel ist das $N'_3 = RLRRRLR = RLR\ R\ RLR = N'_2 R N'_2$.

2 Normalformen für den Kombinator J

Wir betrachten Terme in CL(J), Kombinatorischer Logik mit Kombinator J. Dort haben wir genau zwei Funktionssymbole, ein zweistelliges, das wir als Operator @ schreiben, sowie ein nullstelliges, nämlich J.

Dann ist CL(J) ein Term–Ersetzungs–System mit der einzigen Regel

$$(((\mathrm{J}@a)@b)@c)@d \to (a@b)@((a@d)@c).$$

Nach Vereinbarung lassen wir @ und einige Klammern weg und schreiben kürzer

$$\mathrm{J}abcd \to ab(adc).$$

Bekannterweise ist $\{\mathrm{I}, \mathrm{J}\}$ eine Basis. Umso größer war die Überraschung, als Probst und Studer kürzlich zeigten [PS00], daß CL(J) terminiert. Daraufhin habe ich einige J–Terme untersucht und folgende interessante Familie gefunden:

$$X_0 = \mathrm{J}, X_{k+1} = L[X_k] \textbf{ wobei } L[x] = \mathrm{J}x\mathrm{J}, R[x] = \mathrm{JJ}x$$

Beispielsweise ist $X_3 = \mathrm{J}X_2\mathrm{J} = \ldots = \mathrm{J}(\mathrm{J}(\mathrm{JJJ})\mathrm{J})\mathrm{J}$. Dann gilt der

Satz 1 *Die Normalform von X_k c d ist*

$$\textbf{wenn } 2 \mid k \textbf{ dann } N'_k[\mathrm{J}\ c\ d] \textbf{ sonst } N'_k[\mathrm{J}\ d\ c]$$

Hierbei sind $N'_k \in \{L, R\}^*$ die Wörter aus dem vorigen Abschnitt, diesmal aufgefaßt als ineinandergesetzte Kontexte.
Beispiel: $N'_2 = RLR, N'_2[\mathrm{J}\ c\ d] = R[L[R[\mathrm{J}\ c\ d]]]$.
Der Beweis des Satzes erfolgt leicht per Induktion; im Fall $2 \mid k$ so:

$$X_{k+1}\ c\ d = \mathrm{J}X_k\mathrm{J}\ c\ d \to X_k\mathrm{J}(X_k\ d\ c) \to^* X_k\mathrm{J}\ N'_k[\mathrm{J}\ d\ c]$$
$$\to^* N'_k[\mathrm{JJ}\ N'_k[\mathrm{J}\ d\ c]] = N'_k[R[N'_k[\mathrm{J}\ d\ c]]] = N'_{k+1}[\mathrm{J}\ d\ c]$$

Wir haben hier die Normalformen bezüglich eines Ersetzungsssystems durch iterierte Morphismen beschrieben. Bei beiden Verfahren (Terme ersetzen und Morphismen iterieren) werden Änderungen wiederholt ausgeführt. Der enge Zusammenhang ist jedoch nicht selbstverständlich: Termersetzen ist lokal (jeweils nur an einer Stelle), Morphismus anwenden ist global (jeweils überall). Wir sehen im nächsten Abschnitt ein weiteres Term–Ersetzungs System, bei dem das (anscheinend) funktioniert.
Ich kann bis jetzt nur darüber spekulieren, woran das "wirklich" liegt. Es wird sicher damit zusammenhängen, daß man CL–Terme als Funktionen auffassen sollte, das heißt, eine Betrachtung von J als $\lambda ab.\lambda cd.ab(adc)$ könnte weiterhelfen. In diesem Moment beginnen aber die bekannten Schwierigkeiten der Termersetzung höherer Ordnung.

3 Unendliche Normalformen für CL(S)

Wir betrachten jetzt den Kombinator S mit der Reduktionsregel

$$\mathrm{S}\ x\ y\ z \to x\ z\ (y\ z)$$

Dieses System ist nicht terminierend, aber top–terminierend [Wal99] (und sowieso konfluent). Das bedeutet, daß für Terme X ohne Normalform alle unendlichen (und fairen) Reduktionsketten, die bei X starten, dem gleichen Limes zustreben. Das ist ein unendlicher Baum, den wir die (unendliche) Normalform von X nennen.
Beispiel: Für die Familie $X_0 = \mathrm{S(SS)(SS)}, X_{k+1} = \mathrm{SS}X_k$ gilt

Satz 2 *Kein Term X_iX_k besitzt eine Normalform.*

Beweis: es gilt $X_0X_k \to X_{k+1}X_{k+1}$ und $X_{i+1}X_k \to SX_k(X_iX_k)$.
Hier sehen wir auch die unendliche Normalform von X_0X_0:

$$X_0X_0 \to X_1X_1 \to \mathrm{S}X_1(X_0X_1) \to \mathrm{S}X_1(X_2X_2)$$
$$\to \mathrm{S}X_1(\mathrm{S}X_2(X_1X_2)) \to \mathrm{S}X_1(\mathrm{S}X_2(\mathrm{S}X_2(X_0X_2))) \to \ldots$$

Das ist einfach, weil jeder Term genau einen Redex enthält.
Wir erkennen auf dem rechten Rückgrat die Struktur des quadratischen Wortes aus einem vorigen Beispiel: Es stehen dort ein $\mathrm{S}X_1$, dann zwei $\mathrm{S}X_2$, dann drei $\mathrm{S}X_3$ usw. Es ist deswegen klar, daß dieser unendliche Baum als (homomorphes Bild eines) Fixpunktes eines Morphismus auf Bäumen entsteht.
Ich vermute, daß solch eine Darstellung für unendliche Normalformen beliebiger S–Terme möglich ist. Bisher kann ich das aber nur experimentell bestätigen. Für den Term X_0X_0 scheint das alles sehr einfach zu sein, aber es gibt andere Startterme mit höchst komplizierten Entwicklungen.

4 Vergleichbarkeit von Fixpunkten

Angenommen, es läßt sich beweisen, daß unendliche Normalformen in CL(S) immer durch iterierte Morphismen beschrieben werden können. Dann hätten wir einen Ansatz, um das Wortproblem in CL(S) [Wal00] zu bearbeiten: wegen Konfluenz sind zwei Terme konvertierbar, wenn sie die gleiche (unendliche) Normalform haben. Man bestimmt zu jeder den Morphismus, und muß dann „nur noch“ entscheiden, ob beide Morphismen den gleichen unendlichen Baum erzeugen.

Das ist im Wesentlichen die Frage nach der Äquivalenz von HD0L–Sprachen. Für Wörter ist das lösbar [KRS97], benutzt aber Ehrenfeuchts Vermutung und Makanins Algorithmus. Wie ist das für HD0L–Baumsprachen? Dort besteht wohl im Allgemeinen wenig Aussicht auf Erfolg, zum Beispiel ist die Lösbarkeit von Baumgleichungen bis heute offen [MR98] (Makanins Algorithmus [Die97] löst Wortgleichungen, und ist schon höchst kompliziert). Hoffentlich zeigt sich zwischendurch, daß bei CL(S) nicht beliebige HD0L-Sprachen entstehen, sondern vielleicht nur polynomiell beschränkte. Für diese könnte das Entscheidungsverfahren leichter zu finden sein.

Literatur

[Die97] Volker Diekert. *Makanin's Algorithm*, pages 344–391. In [Lot97a], 1997.

[KRS97] Lila Kari, Grzegorz Rozenberg, and Arto Salomaa. *L Systems*, pages 253 328. Volume 1 of Rozenberg and Salomaa [RS97], 1997.

[Lot97a] M. Lothaire. *Algebraic Combinatorics on Words.* `http://www-igm.univ-mlv.fr/~berstel/Lothaire/`, 1997.

[Lot97b] M. Lothaire. *Combinatorics on Words.* Cambridge Mathematical Library. Cambridge University Press, 1997.

[MR98] Sabrina Mantaci and Antonio Restivo. Codes and equations for trees. Technical Report 194, Turku Centre for Computer Science, August 1998. `http://altair.math.unipa.it/CSG/sabrina/publications.html`.

[PS00] Dieter Probst and Thomas Studer. How To Normalize the Jay. *Theoretical Computer Science*, (to appear), 2000. `http://www.iam.unibe.ch/~tstuder/papers/j.pdf`.

[RS97] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages.* Springer, 1997.

[Wal99] Johannes Waldmann. (Head normalization and) Top termination in CL(S). Talk given at 4th International Workshop on Termination, Schloß Dagstuhl, May 1999, `http://www.informatik.uni-leipzig.de/~joe/talk/wst99-slides.ps`.

[Wal00] Johannes Waldmann. Problem 97: Is the Word Problem for the S Combinator Decidable? RTA list of open problems, `http://www.lri.fr/~rtaloop/97.html`, 2000.

Sublineare Mehrdeutigkeit

Klaus Wich
Institut für Informatik, Universität Stuttgart,
Breitwiesenstr. 20-22, 70565 Stuttgart
E-mail: wich@informatik.uni-stuttgart.de

Zusammenfassung

Eine kontextfreie Grammatik heißt eindeutig, wenn jedes Wort der erzeugten Sprache einen eindeutigen Ableitungsbaum hat. Eine kontextfreie Sprache L heißt inhärent mehrdeutig, wenn es keine eindeutige Grammatik gibt, die L erzeugt. Man kann für mehrdeutige Sprachen und Grammatiken die Anzahl der Ableitungsbäume in Abhängigkeit von der Wortlänge untersuchen. In diesem Vortrag werden erste Beispiele für (linear) kontextfreie Sprachen mit sublinearer Mehrdeutigkeit vorgestellt.

1 Einleitung

Eine kontextfreie Grammatik heißt eindeutig, wenn jedes Wort der erzeugten Sprache einen eindeutigen Ableitungsbaum hat. Eine kontextfreie Sprache heißt inhärent mehrdeutig, wenn sie von keiner eindeutigen Grammatik erzeugt wird. Man kann mehrdeutige Grammatiken und Sprachen nach der Anzahl der Ableitungsbäume differenzieren. Die Anzahl der Ableitungsbäume für ein Wort hängt dabei i.a. von der Wortlänge ab. Jede kontextfreie Grammatik ist entweder exponentiell mehrdeutig oder sie besitzt eine polynomiell beschränkte Mehrdeutigkeit. Bisher gab es nur Beispiele von Grammatiken und Sprachen mit einer Mehrdeutigkeit von $2^{\Theta(n)}$ bzw. $\Theta(n^d)$ für beliebiges $d \in \mathbb{N}_0$. In diesem Vortrag werden erste Beispiele für (linear) kontextfreie Sprachen mit logarithmischer und quadratwurzelförmiger Mehrdeutigkeit vorgestellt.

2 Definitionen

Es wird vorausgesetzt, dass der Leser mit kontextfreien Grammatiken vertraut ist. Wir bezeichnen die Menge der Nichtterminale und Terminale mit N bzw. Σ, das Startsymbol mit S und die Menge der Produktionen mit P. Dabei gilt $P \subseteq N \times (N \cup \Sigma)^*$.

Definition 2.1. *Sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik, $w \in \Sigma^*$ und $n \in \mathbb{N}$. Wir definieren die Mehrdeutigkeit von w und die Mehrdeutigkeits-*

funktion von G wie folgt:

$$\begin{aligned} am_G(w) &:= \textit{Anzahl der Ableitungsbäume für } w. \\ am_G(n) &:= \max\Big\{am_G(w) \mid w \in \Sigma^{\leq n}\Big\} \end{aligned}$$

Definition 2.2. *Sei $f : \mathbb{N}_0 \to \{r \in \mathbb{R} \mid r > 0\}$ eine totale Funktion und L eine kontextfreie Sprache. L heißt inhärent f-deutig, wenn für alle kontextfreien Grammatiken G, die L erzeugen, $am_G = \Omega(f)$ gilt und es eine kontextfreie Grammatik G_0 gibt, die L erzeugt und $am_{G_0} = \mathcal{O}(f)$ erfüllt.*

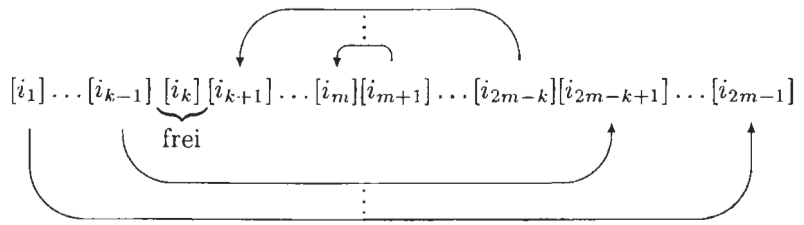
3 Sublineare kontextfreie Sprachen

Definition 3.1. *Sei $\Sigma = \{0,1\}$ und $i \geq 1$. Wir schreiben $[i]$ kurz für $0^{i-1}1$.*

Definition 3.2.

$$\begin{aligned} L_{\log} := \{&[i_1]\dots[i_{2m-1}] \mid m \in \mathbb{N};\ i_1,\dots,i_{2m-1} \in \mathbb{N};\ \exists 1 \leq k \leq m : \\ &((\forall \ell < k : 2i_\ell = i_{2m-\ell}) \wedge (\forall m \geq \ell > k : i_\ell = 2i_{2m+1-\ell}))\} \end{aligned}$$

Sei $w = [i_1]\dots[i_{2m-1}] \in L_{\log}$ mit $i_1,\dots,i_{2m-1} \in \mathbb{N}$ und $m \in \mathbb{N}$. Für $1 \leq k \leq 2m-1$ bezeichnen wir $[i_k]$ als k-ten Block von w. Die Blöcke sind von den Rändern zur Mitte paarweise korreliert. Im Bereich vom ersten bis zum Block in der Mitte gibt es einen Block der mit keinem anderen korreliert sein muß. Wir bezeichnen ihn als freien Block. Hinter diesem Block sind die Blöcke in der umgekehrten Richtung korreliert. Für $L_{\log}$ ist der Quotient der korrelierten Zahlen 2. Will man eine Sprache $L_{\surd}$ mit quadratwurzelförmiger Mehrdeutigkeit definieren, so genügt es statt dem Quotient 2 eine Differenz von 1 für korrelierte Paare zu verlangen. Die gemeinsame Struktur von $L_{\log}$ und $L_{\surd}$ wird im folgenden Diagramm illustriert.



Die Sprachen $L_{\log}$ und $L_{\surd}$ werden von den nachfolgend definierten Grammatiken generiert.

Definition 3.3. $G_{sub} := (\{A,B,C,D,S\},\{0,1\},P_{sub},S)$ *mit:*

$$
\begin{array}{lllll}
P_{\log} := \{ & S & \to & 1S01 \mid 0A01 \mid B & \\
& A & \to & 0A00 \mid 1S00 & \\
& B & \to & 0B \mid 1C \mid 1 & \\
& C & \to & 01C1 \mid 00D1 \mid 011 & \\
& D & \to & 00D0 \mid 01C0 \mid 010 & \}
\end{array}
$$

$$
\begin{array}{lllll}
P_{\surd} := \{ & S & \to & 1S01 \mid 0A1 \mid B & \\
& A & \to & 0A0 \mid 1S00 & \\
& B & \to & 0B \mid 1C \mid 1 & \\
& C & \to & 01C1 \mid 0D1 \mid 011 & \\
& D & \to & 0D0 \mid 01C0 \mid 010 & \}
\end{array}
$$

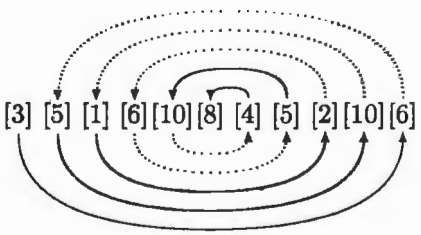
Man sieht leicht, dass die Grammatiken die gewünschten Sprachen generieren. Die Nichtterminale S und A erzeugen dabei die Blöcke links von der freien Position und die zugehörigen Partner. Das Nichtterminal B erzeugt den freien Block, während C und D die übrigen Paare generieren. Im folgenden wird nur noch der logarithmische Fall behandelt. Der quadratwurzelförmige Fall verhält sich völlig analog.

3.1 Sublinear Mehrdeutigkeit der präsentierten Grammatiken

Definition 3.4. *Sei* $i_1, \ldots, i_{2m-1} \in \mathbb{N}$ *mit* $m \in \mathbb{N}$, $w = [i_1] \ldots [i_{2m-1}]$ *und* $1 \leq \ell \leq m$.

- *Das Wort* w *hat eine* Vorwärtskorrelation *am Block* ℓ *genau dann wenn* $\ell < m$ *und* $2i_\ell = i_{2m-\ell}$. *Es hat einen* Vorwärtsbruch *genau dann wenn* $\ell < m$ *und* $2i_\ell \neq i_{2m-\ell}$.
- *Das Wort* w *hat eine* Rückwärtskorrelation *am Block* ℓ *genau dann wenn* $1 < \ell \leq m$ *und* $i_\ell = 2i_{2m+1-\ell}$. *Es hat einen* Rückwärtsbruch *genau dann wenn* $1 < \ell \leq m$ *und* $i_\ell \neq 2i_{2m+1-\ell}$.

Beispiel 3.5. *Wir illustrieren diese Definitionen anhand des folgenden Diagramms. Die für die Betrachtung relevanten Blockpaare findet man auf einer Spirale. Durchgezogene Pfeile stehen für Korrelationen, unterbrochene für Brüche.*



Die Blöcke 1, 2 und 3 haben eine Vorwärtskorrelation. Die Blöcke 4 und 5 haben einen Vorwärtsbruch. Die Blöcke 2, 3 und 4 haben einen Rückwärtsbruch und die Blöcke 5 und 6 haben eine Rückwärtskorrelation.

Definition 3.6. *Seien* $m, r \in \mathbb{N}$, $i_1, \ldots, i_{2m-1} \in \mathbb{N}$ *und* $w = [i_1] \ldots [i_{2m-1}]$.

- $(r * w) := [ri_1] \ldots [ri_{2m-1}]$
- $z_m = \begin{cases} [1] & \text{für } m = 1 \\ [1](4 * z_{m-1})[2] & \text{für } m > 1 \end{cases}$

Zum Beispiel $z_4 = [1][4][16][64][32][8][2]$.

Satz 3.7. *Sei* $i \in \mathbb{N}$. *Das kürzeste Wort mit der Mehrdeutigkeit* i *ist* z_i.

Beweis. Sei w ein beliebiges Wort aus $L_{\log}$ mit einer Mehrdeutigkeit $i \geq 1$. Die Produktion korrelierter Blöcke vollzieht sich in eindeutiger Weise. Die Position des freien Blocks ist hingegen nicht für jedes $w \in L_{\log}$ eindeutig bestimmt. Die Anzahl der Ableitungsbäume für w entspricht gerade der Anzahl der möglichen Positionen für den freien Block. Sei Block Nummer k der am weitesten links stehende Block mit einem Vorwärtsbruch, oder der erste Block, falls keine Vorwärtsbrüche existieren. Dann wissen wir, dass der k-te Block sicher nicht von den Nichtterminalen S und A erzeugt wird. Daraus folgt, dass alle auf der Spirale weiter innen liegenden Blöcke von den Nichtterminalen C und D auf eindeutige Weise erzeugt werden. Deshalb können wir all diese Blöcke unter Erhalt der Mehrdeutigkeit löschen und erhalten so ein Wort w' ohne Vorwärtsbrüche, das höchstens so lang wie w ist und i Ableitungsbäume hat. Sei nun Block Nummer ℓ' in w' der am weitesten rechts stehende Block mit einem Rückwärtsbruch, oder der mittlere Block, falls keine Rückwärtsbrüche existieren. Dann wissen wir, dass der ℓ-te Block sicher nicht von den Nichtterminalen C und D erzeugt wird. Daraus folgt, dass alle auf der Spirale weiter außen liegenden Blöcke von den Nichtterminalen S und A auf eindeutige Weise erzeugt werden. Durch löschen dieser Blöcke erhalten wir ein Wort w''. Wie zuvor bleibt die Mehrdeutigkeit bei dieser Kürzung erhalten. Das Wort w'' ist somit ein bruchfreies Wort mit i Ableitungsbäumen, das höchstens so lang wie w ist. D.h., es gibt $r \geq 1$ so, dass $w'' = (r * z_i)$ gilt. Damit sind in w'' alle Blöcke vom ersten bis zum i-ten Block Kandidaten für den freien Block. Dieselben Überlegungen gelten auch für das Wort z_i. Schließlich erhalten wir $am_G(z_i) = am_G(w'') = am_G(w') = am_G(w) = i$ und $|z_i| \leq |(r * z_i)| = |w''| \leq |w'| \leq |w|$. Durch mehrdeutigkeitserhaltende Längenkürzungen haben wir somit aus einem beliebigen Wort der Mehrdeutigkeit i das Wort z_i erzeugt. □

Die Mehrdeutigkeitsfunktion wird also durch die Wörter z_i für beliebiges $i \geq 1$ dominiert. Die logarithmische Mehrdeutigkeit ergibt sich daher direkt aus der nachfolgenden Tabelle.

Mehrdeutigkeit	kürzestes Wort	Länge
1	$z_1 = [1]$	1
2	$z_2 = [1][4][2]$	7
$\vdots$	$\vdots$	$\vdots$
i	$\ldots$	$\frac{1}{2}4^i - 1$

Satz 3.8. *$L_{\log}$ ist inhärent logarithmisch mehrdeutig.*

Beweisskizze. Sei G eine beliebige kontextfreie Grammatik mit $L(G) = L_{\log}$. Sei p die Konstante von Ogdens Iterationslemma [1, Lemma 2.5] für die Grammatik G. Sei $s := p + 1$ und $r := s! + s$. Wir zeigen die Behauptung indem wir beweisen, dass für ein beliebiges $m \geq 1$ das Wort $(r * z_m)$ mindestens m Ableitungen besitzt. Dazu werden m unterschiedliche Wörter aus $L_{\log}$ gebildet. Diese müssen alle mindestens einen Ableitungsbaum haben. In diesem Baum kann man so pumpen, dass stets das Wort $(r * z_m)$ entsteht. Man kann zeigen, dass die so entstandenen Bäume für $(r * z_m)$ paarweise verschieden sein müssen.

Nun etwas genauer: Zur Bildung der m verschiedenen Ausgangswörter multiplizieren wir einen der ersten m Blöcke von $(r * z_m)$ mit $\frac{s}{r}$. An dieser Stelle entsteht sowohl ein Vorwärts- als auch ein Rückwärtsbruch. Wir markieren die 0-en in diesen Block gemäß Ogdens Lemma. Wir können nun so pumpen, dass eines der beiden gepumpten Teilwörter komplett in der markierten Position liegt. Der andere Teil darf keine 1-en enthalten, denn sonst würden neue Blöcke entstehen, die das Gefüge der Korrelationen durcheinander bringen. Für einen strikten Beweis, dass dies mit der Definition der Sprache nicht vereinbar ist, zeigt man das genügend häufiges Pumpen zu einer genügend weit über den Block in der Mitte hinausragenden 2n-periodischen Blockstrucktur führt, die nicht mit der Sprachdefinition vereinbar ist. Damit können nur 0-en gepumpt werden. D.h die Zahl der Blöcke bleibt beim Pumpen konstant und neben dem markierten Block gibt es höchstens einen weiteren Block, der gepumpt wird. Angenommen es wird tatsächlich in einem weiteren Block gepumpt. Durch einmaliges Pumpen können die Brüche des markierten Blocks nicht „geheilt" werden, in dem weiteren gepumpten Block entstehen aber, im Widerspruch zur Sprachdefinition neue Brüche. Es bleibt die Möglichkeit komplett im markierten Block zu pumpen. Dann kann man wegen der Wahl von r und s nach endlich vielen Pumpschritten das Wort $(r * z_m)$ erzeugen. Würde durch pumpen aus einem anderen der oben genannten m Wörter der gleiche Ableitungsbaum entstehen, so hieße dies, dass wir in diesem Ableitungsbaum in zwei unterschiedlichen Blöcken unabhängig voneinander pumpen könnten. Damit könnten wir uns durch je einen weiteren Pumpschritt in beiden Blöcken aus der Sprache pumpen. Also müssen die durch pumpen aus den m Ausgangswörtern entstandenen Ableitungsbäume für $(r * z_m)$ paarweise verschieden sein.

□

4 Zusammenfassung

Wir haben je eine (linear) kontextfreie Sprache mit logarithmischer und mit quadratwurzelförmiger Mehrdeutigkeit präsentiert. Gibt es sublogarithmische Mehrdeutigkeit? Kann man die möglichen Mehrdeutigkeitsklassen charakterisieren? In [3] wurde eine erste kontextfreie Sprache mit transzendenter Dichte vorgestellt. Sie wurde aus zwei eindeutigen kontextfreien Sprachen entwickelt, deren Durchschnitt eineblogarithmische Zensusfunktion hat. Das heißt, es gab im Schnitt der beiden Sprachen nur $\mathcal{O}(\log n)$ viele Wörter bis zu einer Länge n. Die Vereinigung der Ausgangssprachen bildet eine zweideutige Sprache, die aber nur für logarithmisch viele Wörter zwei Ableitungsbäume benötigt. Durch Konkatenation der Sprachen mit einem dazwischengeschalteten freien Block entsteht eine logarithmisch mehrdeutige kontextfreie Sprache. Durch eine geeignete Permutation der Blöcke entstht die hier vorgestellte linear kontextfreie Sprache. Wir haben also insgesamt die Zensusfunktion des Durchschnitts zweier kontextfreier Sprachen in inhärente Mehrdeutigkeit umgesetzt. Wenn sich diese Methode generalisieren läßt, dann können wir ein ganzes Spektrum weiterer Mehrdeutigkeitsfunktionen erwarten.

Literatur

[1] J. Berstel. *Transductions and Context-Free Languages.* Teubner, 1979.

[2] J.E. Hopcroft, J.D. Ullman. *Introduction to Automata Theory, Formal Languages, and Computation.* Addison-Wesley, 1979.

[3] R. Kemp. *A Note on the Density of Inherently Ambiguous Context-free Languages.* Acta Informatica 14, pp. 295–298, 1980.

[4] M. Naji. *Grad der Mehrdeutigkeit kontextfreier Grammatiken und Sprachen.* Diplomarbeit, FB Informatik, Johann–Wolfgang–Goethe–Universität, Frankfurt am Main, 1998.

[5] K. Wich. *Kriterien für die Mehrdeutigkeit kontextfreier Grammatiken.* Diplomarbeit, FB Informatik, Johann–Wolfgang–Goethe–Universität, Frankfurt am Main, 1997.

[6] K. Wich. *Exponential Ambiguity of Context-free Grammars.* Proc. 4th Int. Conf. on Developments in Language Theory '99, World Scientific, Singapore, to appear.

A Normal Form for Church-Rosser Language Systems

Jens R. Woinowski

Abstract

In [Woi00b] it was shown that under certain conditions it is possible to build systems for prefix languages of Church-Rosser languages. One *seemingly* restrictive condition was that the language is defined by a Church-Rosser language system (CRLS) with rules that look like context-sensitive rules of the form $uvw \to uxw$ with u, v, w being words, w being nonempty and x being a single letter or empty. Because of the aim of constructing a system for the prefix language this property was called *prefix splittable*. A natural question raised was whether prefix splittable Church-Rosser language systems (psCRLS's) are a normal form for CRLS's. In this text, a positive answer is given.

1 Basic definitions

The reader is assumed to be familiar with definitions and notations of confluent string rewriting. For details, see [Jan88], [MNO88], and [BO98].

Definition 1. *A* Church-Rosser language system (CRLS) *is a 6-tuple* $C = (\Gamma, \Sigma, R, k_l, k_r, y)$ *with finite alphabet* Γ*, terminal alphabet* $\Sigma \subset \Gamma$ *(*$\Gamma \setminus \Sigma$ *is the alphabet of nonterminals), finite confluent weight reducing system* $R \subseteq \Gamma^* \times \Gamma^*$*, left and right end marker words* $k_l, k_r \in (\Gamma \setminus \Sigma)^* \cap \mathrm{IRR}(R)$*, and accepting letter* $y \in \Gamma \cap \mathrm{IRR}(R)$*.*

The language defined by C *is:* $L_C := \{w \in \Sigma^* | k_l \cdot w \cdot k_r \to^*_R y\}$

A language L *is called a* Church-Rosser language (CRL) *if there exists a* CRLS C *with* $L_C = L$*.*

The definition of Church-Rosser languages is due to McNaughton, Narendran, and Otto [MNO88].[1] The machine model for CRL is a variant of the deterministic shrinking two pushdown automaton [MNO88].

[1] The definition of Church-Rosser language systems given here is slightly different and using a result of Niemann and Otto [NO97] about weight and length reduction.

Definition 2. *A* CRLS $C = (\Gamma, \Sigma, R, ¢, \$, y)$ *is* prefix splittable (C is a psCRLS) *if* $¢, \$, y \in \mathrm{IRR}(R) \cap \Gamma \setminus \Sigma$ *(let the* inner alphabet be $\Gamma_{inner} := \Gamma \setminus \{¢, \$, y\}$*) and for any rule* $r \in R$ *there exists a splitting* (u, v, w, x) *with:*

1. $r = (uvw, uxw)$.
2. *v is non-empty.*
3. *uvw may contain at most one ¢ and if so at its beginning. Also it can have at most one \$ which only may appear at the end. All other letters of uvw have to be elements of the inner alphabet Γ_{inner}.*
4. *x is a single letter not equal to ¢ or \$ or it is the empty word.*
5. *If v contains ¢ or \$, then $x = y$, u and w are empty, and v is element of the regular language $¢ \cdot \Gamma^*_{inner} \cdot \$$.*
6. *If $x = y$, then u and w are empty, and v is element $¢ \cdot \Gamma^*_{inner} \cdot \$$.*

The splitting (u, v, w, x) of a rule r allowed by this is called potential prefix splitting, *u and w are called* left and right context. *("Potential" because this definition is not unique.)*

2 The normal form theorem

Theorem. *Let $C = (\Gamma, \Sigma, R, k_l, k_r, y)$ be a* CRLS *(without restriction of generality let R be length reducing [NO97]) with language L_C. Then there exists a* psCRLS *C' with $L_{C'} = L_C$.*

Proof. We will give an outline of a construction for the new psCRLS C'.

The construction principles: In order to construct a psCRLS C', we will make use of the following four principles:

1. Our new system will have the property that during the whole reduction process there is always exactly one place in the word where the next reduction rule can be applied.
2. We use a compression alphabet which can store more than one letter of Γ in one letter. This information will be represented by subscripts of the compression letters.
3. These compression letters will be enriched by surplus letters in their subscripts in order to spread weight reductions over more than one letter.
4. Rules of the original system will, in most cases, be simulated by three or four rules in the new system.

The new confluent weight reducing system will be built of five parts R_1 to R_5.

Definition 3. *With Γ being the alphabet of C which consists of all terminal and nonterminal letters, let $\overline{\Gamma}$ be a new alphabet, which is a disjunct copy of Γ. Then $^{-}$ is the bijective morphism that maps Γ into $\overline{\Gamma}$, e.g. a to $\overline{a}$. Let $\sharp$, $\mathbb{c}$, and \$ be new symbols. Let $\Gamma_\sharp := \Gamma \cup \{\sharp\}$ and $\overline{\Gamma}_\sharp := \overline{\Gamma} \cup \{\sharp\}$.*

We define: $W_\sharp := \overline{\Gamma}_\sharp^* \cdot \Gamma_\sharp^* \cap ((\sharp^{\leq 2} \cdot ((\overline{\Gamma} \cup \Gamma) \cdot \sharp\sharp)^* \cdot (\overline{\Gamma} \cup \Gamma) \cdot \sharp^{\leq 2}) \cup \sharp^{\leq 2})$, where $\sharp^{\leq 2}$ is a shorthand for $\{\square, \sharp, \sharp\sharp\}$

Definition 4. *Let $\mu_l = \max\{|u| \mid (u,v) \in R\}$ and $\mu_r = \max\{|v| \mid (u,v) \in R\}$ be the maximum sizes of the respective rule sides. Because R is length reducing $\mu_l > \mu_r$ holds. Let $\mu := \max\{\mu_l, |k_l|, |k_r|\}$. The* compression alphabet Γ_1 *is defined as: $\Gamma_1 := \{\xi_w \mid w \in W_\sharp \text{ wedge } 1 \leq |w| \leq 3\mu + 5\}$. The elements of Γ_1 are called* compression letters. *We can store at least one more letter in a compression letter than the left or right end marker words contain. For distinction, letters in the index of a compression letter will be called* index letters.

Translating the input: The first step in the simulation of C is to translate the input into the compression alphabet. At the same time, we will take care of k_l and k_r. The new end marker letters(!) of C' will be $k_l' := \mathbb{c}$ and $k_r' := \$$.

Short words $w \in L_C, |w| \leq 2$ will be handled separately with a a set R_1 of rules: $R_1 := \{(\mathbb{c}w\$, y) \mid w \in L_C \text{ wedge } |w| \leq 2\}$ Obviously, R_1 can be computed easily.

For translating the input a set of rules R_2 will be used, which works as follows: Decompose k_l and k_r into single letters in the following way: $k_l = a_1 a_2 \cdots a_{|k_l|}$ and $k_r = c_1 c_2 \cdots c_{|k_r|}$. R_2 will be designed to be a suitable confluent and weight reducing rewriting system such that for every $w \in \Sigma^{>2}$ with $w = b_1 b_2 \cdots b_i \cdots b_{|w|}, b_i \in \Sigma (1 \leq i \leq |w|)$ we can make the following reduction with R_2:

$$\mathbb{c}w\$ \rightarrow_{R_2}^* \mathbb{c}\xi_{\overline{a}_1\sharp\sharp\overline{a}_2\sharp\sharp\cdots\overline{a}_{|k_l|}\sharp\sharp\overline{b}_1\sharp\sharp}\xi_{b_2\sharp\sharp}\xi_{b_3\sharp\sharp}\cdots\xi_{b_{|w|-1}\sharp\sharp}\xi_{b_{|w|}\sharp\sharp c_1\sharp\sharp c_2\sharp\sharp\cdots c_{|k_r|}\sharp\sharp}\$$$

where the very last step of the translation shall introduce the first compression letter of the result.

The last overlined letter will mark the position at which the simulation of C will work. In general, the last overlined letter in the subscripts of a compressed word can be identified with the head position of the automata mentioned above. The $\sharp$'s are the *surplus letters* mentioned in the construction principles.

The next step is similar to the shift operations of automata for CRL's. Sometimes it is necessary to *move right* (that is, shift) the position, at which

a reduction might take place. This is done with a further set of rules R_3. Such a shift has to take place whenever the overlined letters within the compression letters form an irreducible string w.r.t. R when the overlining is removed. We ommit the details of R_3, shifting corresponds to overlining the first letter in the indexes which is in Γ. If a shift is necessary but no next such letter without overlining exists no next reduction is possible. Then the simulated system also would have come to an irreducible word.

The "weight spreading" strategy: So far, everything in the construction is technically not very tempting. Although these preliminaries are necessary, we now come to the core of the construction, which will be called *weight spreading*. The idea is to reduce the length of the subscripts of each compression letter changed by at least one.

Example: A rule in R could be $(aaaa, bbb)$. Then one case in the simulation would be:

$$\xi_{\overline{a}\sharp\sharp\overline{a}}\xi_{\sharp\sharp\overline{a}\sharp\sharp}\xi_{\overline{a}\sharp\sharp a}$$

$\longrightarrow$ 1. "lock" with new nonterminal

$$\xi_{\overline{a}\sharp\sharp\overline{a}}\xi_{\sharp\sharp\overline{a}\sharp\sharp}\xi_{t}$$

$\longrightarrow$ 2. change first letter

$$\xi_{\overline{b}\sharp\sharp}\xi_{\sharp\sharp\overline{a}\sharp\sharp}\xi_{t}$$

$\longrightarrow$ 3. change middle letter

$$\xi_{\overline{b}\sharp\sharp}\xi_{b\sharp\sharp b}\xi_{t}$$

$\longrightarrow$ 4. change last letter, remove lock

$$\xi_{\overline{b}\sharp\sharp}\xi_{b\sharp\sharp b}\xi_{\sharp\sharp a}$$

The $\sharp$'s are used to spread the length reduction of the original rule over the compression letters in the simulation. Observe that (especially) in the last step of the example the result has three compression letters. This can always be achieved during any rule simulation, because a right rule side and the added $\sharp$'s fit into one compression letter (to be exact, sometimes some *unchanged context* appears on the right). Basically, the weight of a compression letter will be computed from the number of its index letters (overlined letters add *slightly* less to the weight). Therefore in the worst case --when the original rule has a length reduction of one—we reduce the count of index letters by three and spread this reduction over the resulting three compression letters. This is the cause for adding two $\sharp$'s after each index letter of $\Gamma \cup \overline{\Gamma}$ during the translation with R_2.

In generalising this example, a lot of cases have to be handled. The main idea is to identify all possibilities how the left side of an original rule can be divided over one or more letters of the compression alphabet. Also, sometimes it is necessary to allow some unchanged context in the first compression letter. Furthermore, the question of finding the right place for the

reduction to work has to be handled.

After identifying all possible cases, we get a set of new rules that work similar to the example above. These will build R_4. Also the example shows a letter ξ_t. Just like in this case, in some cases further nonterminals are necessary. They are collected in an alphabet Γ_2. For details see [Woi00a].
The last steps: After achieving this, some final rules will be necessary which end accepting reductions. They will be collected in R_5.

The resulting CRLS C' is defined as follows:

Let $\Gamma' := \Sigma \cup \{\mathfrak{c}, \$, y\} \cup \Gamma_1 \cup \Gamma_2$, $\Sigma' := \Sigma$, $R' := R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5$. Then $C' := (\Gamma', \Sigma', R', \mathfrak{c}, \$, y)$ is a psCRLS and $L_{C'} = L_C$.
Correctness of the simulation: Finally, the correctness of the construction has to be proved. Mainly this covers the questions of weight reduction, confluence, and language inclusion and is quite straight forward.

3 Conclusion

The normal form theorem shows that the property of prefix splittablity is no restriction. Concerning prefix systems for CRL's described in [Woi00b] this raises some questions about the decidability of their correctness. Additionally, it should be possible to expand the result to general growing context-sensitive languages and even to their machine model.

References

[BO98] G. Buntrock and F. Otto. Growing context-sensitive languages and Church-Rosser languages. *Information and Computation*, 141:1–36, 1998.

[Jan88] M. Jantzen. *Confluent String Rewriting.* Springer-Verlag, 1988.

[MNO88] R. McNaughton, P. Narendran, and F. Otto. Church-Rosser Thue systems and formal languages. *Journal Association Computing Machinery*, 35:324–344, 1988.

[NO97] G. Niemann and F. Otto. The Church-Rosser languages are the deterministic variants of the growing context-sensitive languages. Technical Report GhK-FB 17: 8/97, Universitt Gesamthochschule Kassel, 1997.

[Woi00a] J. R. Woinowski. A normal form for Church-Rosser language systems. Report, TU-Darmstadt, www.iti.tu-darmstadt.de/~woinowsk/, June 2000.

[Woi00b] J. R. Woinowski. Prefixes of Church-Rosser languages. Report TI-2/00, TU-Darmstadt, www.iti.tu-darmstadt.de/~woinowsk/, February 2000.