PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 42, NO. 1, PP. 71-78 (1998)

# MOLECULAR COMPUTING WITH TEST TUBE SYSTEMS

Rudolf FREUND

Institut für Computersprachen
Technische Universität Wien
Resselgasse 3
A-1040 Wien, Austria
email: rudi@logic.tuwien.ac.at
telephone: + 43 1 588 01 4084
fax: + 43 1 504 1589

Received: Oct. 20, 1997

## Abstract

In this paper a survey of various different theoretical models of test tube systems is given. In test tube systems specific operations are applied to the objects in their components (test tubes) in a distributed and parallel manner; the results of these computations are redistributed according to a given output/input relation using specific filters. A general theoretical framework for test tube systems is presented which is not only a theoretical basis of systems used for practical applications, but also covers the theoretical models of test tube systems based on the splicing operation as well as of test tube systems based on the operations of cutting and recombination. For test tube systems based on the operations of cutting and recombination we show that in one test tube from a finite set of axioms and with a finite set of cutting and recombination rules only regular languages can evolve.

*Keywords:* molecular computing, splicing, test tubes.

## 1. Introduction

Test tube systems were introduced as biological computer systems based on DNA molecules ([1], [2], [3], [11]), and the practical solution of various problems (e.g. even of NP complete problems like the Hamiltonian path problem in [1]) with such systems was described. The theoretical features of test tube systems based on the splicing operation were investigated in [4]; in [7] test tube systems based on the operations of cutting and recombination were explored; in both cases, these test tube systems were shown to have the computational power of Turing machines. A general theoretical framework for test tube systems was described in [8].

In the second section we define the notions from formal language theory needed in this paper and we introduce the formal definitions for the general models of test tube systems described in [8] as well as the notions of test tube systems based on the splicing operation defined in [4] and the test tube

systems based on the operations of cutting and recombination described in [7]. In the third section we recall some results obtained for test tube systems based on the splicing operation from [4] and for test tube systems based on the operations of cutting and recombination from [7]; moreover, we prove that extended cutting/recombination systems with a finite set of axioms and a finite set of rules exactly represent the regular languages, a question which was left open in [7] (in other words this shows that in one test tube from a finite set of axioms and with a finite set of cutting and recombination rules only regular languages can evolve). A short overview of related research topics concludes the paper.

## 2. Definitions

In this section we define some notions from formal language theory and recall the definitions of splicing schemes (H-schemes; see [4], [5], [10], [12], [13]) and of cutting/recombination schemes (CR-schemes; confer to [6]). Moreover, we introduce the definitions for the general theoretical model of test tube systems with prescribed output/input relations as well as for test tube systems based on the splicing operation from [4] and for test tube systems based on the operations of cutting and recombination from [7].

### *2.1. Formal Language Theory Prerequisites*

In this subsection we only define some notions from formal language theory that we shall need in this paper.

The free monoid generated by the alphabet $V$ is denoted by $V^*$, its elements are called *strings* or *words* over $V$; $\lambda$ is the empty string, $V^+ = V^* \setminus \{\lambda\}$.

A *grammar scheme* $\gamma$ is a triple $(V_N, V_T, P)$, where $V_N$ is a (finite) alphabet of *non-terminal symbols;* $V_T$ is a (finite) alphabet of *terminal symbols* with $V_N \cap V_T = \emptyset$; $P$ is a (finite) set of *productions* of the form $(\alpha, \beta)$, where $\alpha \in (V_N \cup V_T)^+$ and $\beta \in (V_N \cup V_T)^*$. For two words $x, y \in (V_N \cup V_T)^+$, the *derivation relation* $\vdash_\gamma$ is defined if and only if $x = u\alpha v$ and $y = u\beta v$ for some production $(\alpha, \beta) \in P$ and two strings $u, v \in (V_N \cup V_T)^*$; we then also write $x \vdash_\gamma y$. The reflexive and transitive closure of the relation $\vdash_\gamma$ is denoted by $\vdash_\gamma^*$. A *grammar* $G$ is a quadruple $(V_N, V_T, P, S)$, where $\gamma = (V_N, V_T, P)$ is a grammar scheme and $S \in V_N$. The $\lambda$-free language generated by G is $L(G) = \left\{ w \in V_T^+ \mid S \vdash_\gamma^* w \right\}$. The grammar $G$ is called *regular,* if every production in $P$ is of the form $(A, w)$, where $A \in V_N$ and $w \in V_T V_N \cup V_T$.

The family of ($\lambda$-free) languages generated by arbitrary and regular grammars is denoted by $ENUM$ and $REG$, respectively, and the family of

finite ($\lambda$-free) languages is denoted by $FIN$. By $REG^+$ we denote the family of regular languages of the form $W^+$ for some finite set $W$.

### 2.2. Splicing Schemes and Cutting/Recombination Schemes

We now recall the definitions of splicing schemes (H-schemes; see [4], [5], [10], [12], [13]) and of cutting/recombination schemes (CR-schemes; confer to [6]).

As the empty word has no meaningful representation in nature, $\lambda$ is not considered to be an object we have to deal with; as for grammars above, also in the following only mechanisms for generating $\lambda$-free languages will be considered (all the definitions we shall give have been adapted in a suitable manner).

A *splicing scheme (H-scheme)* is a pair $\sigma$, $\sigma = (V, R)$, where $V$ is an alphabet and $R \subseteq V^*\#V^*\$V^*\#V^*$; $\#, \$$ are special symbols not in $V$. $R$ is the set of *splicing rules*. For $x, y, z \in V^+$ and $r = u_1\#u_2\$u_3\#u_4$ in $R$ we define $(x, y) \vdash_r z$ if and only if $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, and $z = x_1u_1u_4y_2$ for some $x_1, x_2, y_1, y_2 \in V^*$.

For any language $L \subseteq V^+$, we write

$$\sigma(L) = \{z \in V^+ \mid (x, y) \vdash_r z \text{ for some } x, y \in L, r \in R\},$$

and we define $\sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L)$, where
$\sigma^0(L) = L$, $\sigma^{i+1}(L) = \sigma(\sigma^i(L))$ for $i \geq 0$.

An *extended H-system* (or *extended splicing system*) is a quadruple $\gamma$, $\gamma = (V, V_T, A, R)$, where $V_T \subseteq V$ is the set of terminal symbols and $A$ is the set of axioms. The *language generated* by the extended H-system $\gamma$ is defined by $L(\gamma) = \sigma^*(A) \cap V_T^+$.

A *cutting/recombination scheme* (or a *CR-scheme*) is a quadruple $\sigma = (V, M, C, R)$, where $V$ is a finite alphabet; $M$ is a finite set of *markings;* $V$ and $M$ are disjoint sets; $C$ is a set of *cutting rules* of the form $u\#l\$m\#v$, where $u \in V^* \cup MV^*$, $v \in V^* \cup V^*M$, and $m, l \in M$, and $\#, \$$ are special symbols not in $V \cup M$; $R \subseteq M \times M$ is the recombination relation representing the *recombination rules.* Cutting and recombination rules are applied to objects from $O(V, M)$, where we define

$$O(V, M) = V^+ \cup MV^* \cup V^*M \cup MV^*M.$$

For $x, y, z \in O(V, M)$ and a cutting rule $c = u\#l\$m\#v$ we define $x \vdash_c (y, z)$ if and only if for some $\alpha \in V^* \cup MV^*$ and $\beta \in V^* \cup V^*M$ we have $x = \alpha uv\beta$ and $y = \alpha ul$, $z = mv\beta$. For $x, y, z \in O(V, M)$ and a recombination rule $r = (l, m)$ from $R$ we define $(x, y) \vdash_r z$ if and only if for some $\alpha \in V^* \cup MV^*$ and $\beta \in V^* \cup V^*M$ we have $x = \alpha l$, $y = m\beta$, and $z = \alpha\beta$. For a CR-scheme $\sigma = (V, M, C, R)$ and any language $L \subseteq O(V, M)$ we write

$$\sigma(L) = \{y \mid x \vdash_c (y,z) \text{ or } x \vdash_c (z,y) \text{ for some } x \in L, c \in C\} \cup \{z \mid (x,y) \vdash_r z \text{ for some } x,y \in L, r \in R\};$$

$\sigma^*(L)$ and $\sigma^i(L)$ for $i \geq 0$ are defined in a similar way as for splicing schemes.

An *extended CR-system* is a sextuple $\gamma, \gamma = (V, M, V_T, A, C, R)$, where $V_T \subseteq V$ is the set of terminal symbols, $A \subseteq O(V, M)$ is the set of axioms, and $(V, M, C, R)$ is the underlying CR-scheme. The *language generated* by the extended CR-system $\gamma$ is defined by $L(\gamma) = \sigma^*(A) \cap V_T^+$.

Thus $\sigma(L)$ contains all objects obtained by applying one cutting or one recombination rule to objects from $L$; $\sigma^*(L)$ is the smallest subset of $O(V, M)$ that contains $L$ and is closed under the cutting and recombination rules of $\sigma$. $L(\gamma)$ is the set of all terminal words that can be obtained from the axioms by an arbitrary number of cuttings and recombinations.

In [12] it was shown that H-systems with a finite set of axioms and a finite set of splicing rules characterize $REG$, whereas with a regular set of splicing rules we obtain $ENUM$. In [5] it was proved that by adding specific control mechanisms like multisets or context conditions (permitting and forbidden contexts, respectively) to extended H-systems with a finite number of axioms and a finite number of splicing rules again the computational power of Turing machines or arbitrary grammars can be obtained. Similar results for CR-systems were proved in [6]; yet the question concerning the computational power of extended CR-systems with a finite number of axioms and a finite number of cutting and recombination rules was not proved there and will be shown in the following section.

### 2.3. *Test Tube Systems*

In this subsection we recall the definitions for several models of test tube systems defined in [8] as well as [4] and [7].

A *test tube system with prescribed output/input relations* (a *TTSPOI* for short) $\sigma$ is a quintuple $(B, n, A, \rho, D)$, where

1. $B$ is a set of *objects;*
2. $n$, $n \geq 1$, is the number of test tubes in $\sigma$;
3. $A = (A_1, ..., A_n)$ is a sequence of sets of *axioms*, where $A_i \subseteq B$, $1 \leq i \leq n$;
4. $\rho$ is a sequence $(\rho_1, ..., \rho_n)$ of sets of *test tube operations*, where $\rho_i$ contains specific operations for the test tube $T_i$, $1 \leq i \leq n$;
5. $D$ is a (finite) set of *prescribed output/input relations* between the test tubes in $\sigma$ of the form $(i, F, j)$, where $1 \leq i \leq n$, $1 \leq j \leq n$ and $F$ is a (recursive) subset of $B$; $F$ is called a filter between the test tubes $T_i$ and $T_j$.

In order to indicate the number of test tubes, we also say that $\sigma$ is a TTSPOI$_n$.

The computations in the system $\sigma$ run as follows: At the beginning of the computation the axioms are distributed over the $n$ test tubes according to $A$, i.e. test tube $T_i$ starts with $A_i$. Now let $L_i$ be the contents of test tube $T_i$ at the beginning of a derivation step. Then in each test tube the rules of $\rho_i$ operate on $L_i$, i.e. we obtain $\rho_i^* (L_i)$. The next substep is the redistribution of the $\rho_i^* (L_i)$ over all test tubes according to the corresponding output/input relations from $D$, i.e. if $(i, F, j) \in D$, then the test tube $T_j$ from $\rho_i^* (L_i)$ gets $\rho_i^* (L_i) \cap F$, whereas the rest of $\rho_i^* (L_i)$ that cannot be distributed to other test tubes remains in $T_i$. The final result of the computations in $\sigma$, $L(\sigma)$, consists of all objects from $B$ that can be extracted from the *final test tube* $T_1$; hence usually we shall assume $F = \emptyset$ for all $(1, F, j) \in D$, as well as $\rho_1 = \emptyset$. Moreover, we say that $\sigma$ is of type $(F_1, F_2, F_3)$, if $A_i \in F_1$, $\rho_i \in F_2$ for all $i$ with $1 \leq i \leq n$, and $F \in F_3$ for all $F$ with $(i, F, j) \in D$ for some $i, j$ with $1 \leq i \leq n$, $1 \leq j \leq n$.

Special variants of this general model have already been formalized for the splicing operation in [4] and for the operations of cutting and recombination in [7].

A *CR-TTSPOI* $\sigma$ is a TTSPOI $(O(V, M), n, A, \rho, D)$, where $\rho = (\rho_1, ..., \rho_n)$, $\rho_i = (C_i, R_i)$, $1 \leq i \leq n$, and $\sigma_i = (V, M, C_i, R_i)$ is a CR-scheme; $L(\sigma)$ usually is taken to be a subset of $V^+$. An H-TTSPOI $\sigma$ is a TTSPOI $(V^+, n, A, \rho, D)$ where $\sigma_i = (V, \rho_i)$, $1 \leq i \leq n$, is an H-scheme.

Whereas in an H-TTSPOI usually filters from $REG$ or even $REG^+$ only are needed, in a CR-TTSPOI usually the following kinds of filters are used:

A subset of $O(V, M)$ is called a *simple* $(V, M)_2$*-filter* if it equals

1. $V^+$ or
2. $\{m\} V^*$ for some $m \in M$ or
3. $V^* \{m\}$ for some $m \in M$ or
4. $\{m\} V^* \{n\}$ for some $m, n \in M$.

A simple $(V, M)_2$-filter is called a *simple* $(V, M)_1$*-filter*, if it is not of the form $\{m\} V^* \{n\}$. Any finite union of simple $(V, M)_i$-filters, $i \in \{1, 2\}$, is called a $(V, M)_i$*-filter;* the families of $(V, M)_i$-filters and simple $(V, M)_i$-filters for arbitrary $V, M$ are denoted by $CRF_i$ and $CRSF_i$, respectively.

## 3. Results

The following results were established in [4], [7], and [8]:

THEOREM 1 *For every* $L \in ENUM$, $L \subseteq V_T$, *we can construct an H-TTSPOI*$_{8+card(V_T)}$ *of type* $(FIN, FIN, REG^+)$ *which generates* $L$.

In fact, the best result known so far and communicated by Gheorghe Păun says that an H-TTSPOI$_7$ is already sufficient, i.e. the number of test tubes can be bounded.

THEOREM 2 *For every $L \in ENUM$ we can construct a CR-TTSPOI of type $(FIN, FIN, CRF_1)$ which generates $L$.*

For the CR TTSPOI in Theorem 2 it is an open question whether the number of test tubes needed for generating arbitrary recursively enumerable languages can be bounded or not.

THEOREM 3 *For every $L \in ENUM$ we can construct a CR-TTSPOI of type $(FIN, FIN, CRSF_i)$, $i \in \{1, 2\}$, which generates $L$:*

THEOREM 4 *For every $L \in ENUM$ we can construct a CR-TTSPOI$_4$ of type $(FIN, FIN, CRF_2)$ which generates $L$.*

For obtaining universal computability, the number of test tubes $n$ in a CR-TTSPOI$_n$ of type $(FIN, FIN, CRF_2)$ may already be optimal with being four; in any case, this number cannot be reduced to less than three:

- Any language generated by a CR-TTSPOI$_1$ of type $(FIN, FIN, CRF_2)$ is finite, because according to the definitions given above it equals the set of axioms in the single test tube.
- Any language generated by a CR-TTSPOI$_2$ of type $(FIN, FIN, CRF_2)$ is regular, because in one test tube only a regular language can evolve as shown in the following theorem.

THEOREM 5 *The family of languages generated by extended CR-systems with finite sets of axioms and finite sets of cutting and recombination rules equals the family of regular languages.*

*Proof.* a) Let $\gamma$, $\gamma = (V, M, V_T, A, C, R)$, be an extended CR-system with a finite set of axioms $A$ and finite sets of cutting and recombination rules $C$ and $R$. We now construct an extended splicing system $\sigma$ such that $L(\sigma) = L(\gamma)$.

According to the results proved in [13] and [14], $L(\sigma)$ is a regular language, hence $L(\gamma)$ is a regular language, too:

The extended splicing system $\sigma$ is defined by

$\sigma = (V \cup M \cup \{Y, Z\}, V_T, A', R_1 \cup R_2)$,

where $Y$ and $Z$ are new symbols not in $V \cup M$,

$$A' = \tilde{A} \cup \{YmZ \mid m \in M\},$$

$$\tilde{A} = (V^* \cap A) \cup \{Y\}(MV^* \cap A) \cup (V^*M \cap A) \cup \{Z\} \cup \{Y\}(MV^*M \cap A)\{Z\},$$

$$R_1 = \{u\#v\$Y\#mZ, Yn\#Z\$u\#v \mid u\#m\$n\#v \in C\}, \text{ and}$$

$$R_2 = \{\#mZ\$Yn\# \mid (m, n) \in R\}.$$

The splicing rules in $R_1$ simulate the cutting rules of $\gamma$, whereas the splicing rules in $R_2$ simulate the recombination rules of $\gamma$. The axioms as well as the objects derived by $\sigma$ are constructed in such a way that a left (right) end marker $m$ in an object derived by $\gamma$ corresponds with a left (right) end marker $Ym$ ($mZ$, respectively) in an object derived by $\sigma$, i.e. the object $w$ in $\sigma$ corresponds with the object $h(w)$ in $\gamma$, where $h : V \cup M \cup \{Y, Z\} \to V \cup M$ is the projection with $h(Y) = h(Z) = \lambda$ and $h(X) = X$ for all $X \in V \cup M$.

For example, given two cutting rules $u_1\#m_1\$n_1\#v_1$ and $u_2\#m_2\$n_2\#v_2$ in $C$ as well as two objects $h(x_1u_1v_1y_1)$ and $h(x_2u_2v_2y_2)$ in $O(V, M)$, we obtain two new objects $x_1u_1m_1Z$ and $Yn_2v_2y_2$ derived in $\sigma$ by using the splicing rules $u_1\#v_1\$Y\#m_1Z$ and $Yn_2\#Z\$u_2\#v_2$ from $R_1$. Such objects $x_1u_1m_1Z$ and $Yn_2v_2y_2$ can be recombined according to the splicing rule $\#m_1Z\$Yn_2\#$ from $R_2$ yielding the object $x_1u_1v_2y_2$ in $\sigma$ in the same way as the two objects $h(x_1u_1m_1Z)$ and $h(Yn_2v_2y_2)$ yield the corresponding object $h(x_1u_1v_2y_2)$ in $\gamma$ by using the recombination rule $(m_1, n_2)$ from $R$ in $\gamma$.

Finally it should be stated that parasitic strings like $YZ$ additionally evolving in $\sigma$ have no influence on the final result $L(\sigma)$.

b) On the other hand, let $G$ be a regular grammar, $G = (V_N, V_T, P, S)$. Then $L(\gamma) = L(G)$ for the CR-system $\gamma$ with

$\gamma = (V_T, \{X^+, X^- \mid X \in V_N\}, A, \emptyset, R)$,
$R = \{(X^+, X^-) \mid X \in V_N\}$, and
$A = \{S^+\} \cup \{Y^-aX^+ \mid Y \to aX \in P\} \cup \{Y^-a \mid Y \to a \in P\}$.

By using the appropriate axioms from $A$ and suitable recombination rules from $R$, for arbitrary words $w \in V_T^*$, $X, Y \in V_N$, and $a \in V_T$, a derivation step in $G$

$wY \vdash_G waX$ (or $wY \vdash_G wa$)

corresponds with the derivation step in $\gamma$

$(wY^+, Y^-aX^+) \vdash_\gamma waX^+$ (or $(wY^+, Y^-a) \vdash_\gamma wa$, respectively).

Hence, in the CR-system $\gamma$ there is no need for extended symbols or for cutting rules; the markers take over the roles of the non-terminal symbols in the regular grammar $G$.

## 4. Conclusion

In this paper we presented various theoretical models of test tube systems and gave an overview of results shown in [4], [7], and [8].

In [9] test tube systems with controlled applications of rules were introduced, and several variants of test tube systems with controlled applications of cutting and recombination rules were shown to have universal computational power.

Many of the results exhibited in this paper also hold true for test tube systems working on a mixture of linear as well as circular strings (confer to [7], [14], and [15]).

## Acknowledgements

The author gratefully appreciates fruitful discussions with Erzsébet Csuhaj-Varjú, Franziska Freund, and Gheorghe Păun on some of the topics considered in this paper.

## References

[1] ADLEMAN, L. M.: Molecular Computation of Solutions to Combinatorial Problems, *Science*, Vol. 226 (Nov. 1994), pp. 1021 – 1024.

[2] ADLEMAN, L. M.: On Constructing a Molecular Computer, manuscript, January 1995.

[3] BONEH, D. – DUNWORTH, C. – LIPTON, R.J. SGALL, J. : On the Computational Power of DNA, to appear.

[4] CSUHAJ-VARJÚ, E. – KARI, L. – PĂUN, GH.: Test Tube Distributed Systems Based on Splicing, *Computers and Artificial Intelligence*, Vol. 15 (2) (1996), pp. 211–232.

[5] FREUND, R. – KARI, L. – PĂUN, GH. DNA Computing Based on Splicing: The Existence of Universal Computers, *Techn. Report* 185-2/FR-2/95, TU Wien, 1995.

[6] FREUND, R. – WACHTLER, F.: Universal Systems with Operations Related to Splicing, *Computers and Artificial Intelligence*, Vol. 15 (4) (1996), pp. 273–294.

[7] FREUND, R. – CSUHAJ-VARJÚ, E. – WACHTLER, F.: Test Tube Systems with Cutting/Recombination Operations, *Proceedings PSB'97*, World Scientific (1997), pp. 163–174.

[8] FREUND, R. – FREUND, F.: Test Tube Systems or How to Bake a DNA Cake, *Acta Cybernetica* Vol. 12, (1996), pp. 445–459.

[9] FREUND, R. – FREUND, F.: Test Tube Systems with Controlled Applications of Rules, *Proceedings ICEC'97*, Indianapolis, Indiana, USA, April 1997.

[10] HEAD, T.: Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors, *Bull. Math. Biology*, Vol. 49 (1987), pp. 737–759.

[11] LIPTON, R. J.: Speeding up Computations via Molecular Biology, manuscript, December 1994.

[12] PĂUN, GH.: Regular Extended H Systems are Computationally Universal, *J. of Automata, Languages and Combinatorics*, Vol. 1, Nr. 1 (1996), pp. 27–37.

[13] PIXTON, D.: Regularity of Splicing Languages, *Discrete Applied Mathematics* Vol. 69 (1996), pp. 101-124.

[14] PIXTON, D.: Splicing in Abstract Families of Languages, manuscript, 1997.

[15] SIROMONEY, R. – SUBRAMANIAN, K. G. – DARE, V. R.: Circular DNA and Splicing Systems, *Lecture Notes in Computer Science*, Vol. 654, Springer-Verlag, Berlin (1992), pp. 260–273.