Acta Cybernetica **12** (1996) 445-459.

Test Tube Systems or How to Bake a DNA Cake

Rudolf FREUND * Franziska FREUND †

Abstract

We introduce various general models for test tube systems which not only are a theoretical basis for the different test tube systems used for practical applications (confer to [1], [2], [3], [12]), but also cover different theoretical models to be found in literature, e.g. the test tube systems based on the splicing operation introduced in [4] as well as test tube systems based on the operations of cutting and recombination introduced in [9]. 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 specific input and/or output filters. We investigate relations between the different models of test tube systems introduced in this paper and show how the results presented in [4] and [9] fit into our general framework. Moreover, we investigate the computational power of test tube systems with context-free productions and regular filters.

1 Introduction

Test tube systems were introduced as biological computer systems based on DNA molecules ([1], [2], [3], [12]), 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 [9] 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.

Most of the test tube systems to be found in literature have the following common features: in the components (test tubes) of the systems specific operations are applied to the objects in the tubes in a distributed and parallel manner; the results of these computations are redistributed according to specific output and/or input filters which only allow specific parts of the contents of a tube to pass over to other test tubes. As it was shown in the theoretical papers mentioned above ([4], [9]), even very restricted kinds of such filters testing for the existence respectively non-existence of specific symbols respectively markings allow for reaching the

*Institut für Computersprachen, Technische Universität Wien Resselgasse 3, A-1040 Wien, Austria. Email: rudi@logic.tuwien.ac.at

†Gymnasium der Schulbrüder, Strebersdorf, Anton Böck-Gasse 37, A-1215 Wien, Austria. Email: freund@strebersdorf.ac.at

computational power of Turing machines. The computational universality of these specific variants of test tube systems was proved in these papers, too.

In [5] and [6] a general framework for describing networks of language identifying devices (networks of language processors) was introduced and investigated. In this paper we shall restrict ourselves to introduce various general models for test tube systems. We investigate relations between these models of test tube systems and also consider their computational power with respect to the complexity of the output/input relations and the filters we use. Moreover, we show how the results presented in [4] and [9] fit into the general framework presented in this paper.

In the second section we start with defining the notions from formal language theory needed in this paper; we introduce the formal definitions for the general models of test tube systems to be investigated in this paper as well as the notions of test tube systems based on the splicing operation ([4]) and the test tube systems based on the operations of cutting and recombination ([9]); moreover, we give some examples illustrating the notions of test tube systems and we show how the test tube systems based on the splicing operation ([4]) and the test tube systems based on the operations of cutting and recombination ([9]) can be interpreted in the framework introduced in this paper. In the third section we investigate some characteristic features of the different general models of test tube systems and also elaborate some specific results for test tube systems which use context-free productions in the test tubes and regular filters for redistribution. A short summary of the results obtained in this paper and an overview of future research topics conclude the paper.

2 Definitions and Examples

In this section we define some notions from formal language theory and recall the definitions of splicing schemes (H-schemes; see [4], [7], [13]) and of cutting/recombination schemes (CR schemes; confer to [8]). Moreover, we introduce the general definitions of test tube systems and give some explanatory examples.

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. For general formal language theory prerequisites we refer to [16].

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

A *grammar scheme* γ 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 (α, β), 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 $S \in V_N$; in a more general way, we can also take $S \in (V_N \cup V_T)^+$. The λ-free language generated by G is $L(G) = \{w \in V_T^+ \mid S \vdash_\gamma^* w\}$.

The grammar G, $G = (V_N, V_T, P, S)$, as well as the corresponding grammar scheme (V_N, V_T, P) is called *context-free,* if every production in P is of the form (A, w), where $A \in V_N$ and $w \in (V_N \cup V_T)^*$; 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 (λ-free) languages over V_T generated by arbitrary, context-free, and regular grammars is denoted by $ENUM(V_T)$, $CF(V_T)$, and $REG(V_T)$, respectively, and the family of finite (λ-free) languages over V_T is denoted by $FIN(V_T)$. The corresponding families of languages over arbitrary terminal alphabets are denoted by $ENUM$, CF, REG, and FIN, respectively. By REG^+ we denote the family of regular languages of the form W^+ for some finite set W.

A *grammar scheme* $\gamma_U(V_T)$ with $\gamma_U(V_T) = (V_N, V_T, P)$ is called *universal (for* V_T*)* if for every $L \in ENUM(V_T)$ there exists a word A_L such that the grammar G_L with $G_L = (V_N, V_T, P, A_L)$ generates L. One of the important results of formal language theory is that for every V_T such a universal grammar $\gamma_U(V_T)$ exists.

2.2 Splicing schemes and cutting/recombination schemes

We now recall the definitions of splicing schemes (H-schemes; see [4], [7], [13]) and of cutting/recombination schemes (CR-schemes; confer to [8]).

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

Definition 1. A *splicing scheme (H-scheme)* is a pair σ, $\sigma = (V, R)$, 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 $x, y, z, w \in V^+$ and $r = u_1\#u_2\$u_3\#u_4$ in R we define $(x, y) \vdash_r (z, w)$ if and only if $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, and $z = x_1u_1u_4y_2$, $w = y_1u_3u_2x_2$, for some $x_1, x_2, y_1, y_2 \in V^*$.

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

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

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

An *extended H-system* (or *extended splicing system*) is a quadruple γ, $\gamma = (V, V_T, A, R)$, 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 γ is defined by $L(\gamma) = \sigma^*(A) \cap V_T^+$.

Definition 2. 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

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

$\sigma^*(L)$ 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 γ, $\gamma = (V, M, V_T, A, C, R)$, where $V_T \subseteq V$ is the set of terminal symbols, A is the set of axioms, and (V, M, C, R) is the underlying CR-scheme. The *language generated* by the extended CR-system γ 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 σ. $L(\gamma)$ is the set of all terminal words that can be obtained from the axioms by an arbitrary number of cuttings and recombinations.

There is a close relationship between CR-schemes and splicing schemes (H-schemes): For instance, applying the splicing rule $u_1\#u_2\$u_3\#u_4$ to two strings $x_1u_1u_2x_2$ and $y_1u_3u_4y_2$ yields the two strings $x_1u_1u_4y_2$ and $y_1u_3u_2x_2$ which corresponds to cutting the strings $x_1u_1u_2x_2$ and $y_1u_3u_4y_2$ into the strings $x_1u_1[m]^+$, $[m]^- u_2x_2$ and $y_1u_3[m]^+$, $[m]^- u_4y_2$ by using the cutting rules $u_1\#[m]^+\$[m]^-\#u_2$ and $u_3\#[m]^+\$[m]^-\#u_4$ and recombining them immediately by applying the recombination rule $\left([m]^+, [m]^-\right)$ in a crosswise way.

In [13] 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 [7] it was proved that by adding specific control mechanisms like multisets or context conditions (permitting respectively forbidden contexts) 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 [8].

2.3 Test tube systems

In this section we introduce several general models of test tube systems (confer to [2], [3], [4], [12] for practical implementations). The idea of test tube systems is to

describe computational devices where the computations in each test tube are based on specific operations and where any computation is done in a distributed way. As a communication step the resulting contents of the test tubes then is redistributed according to specific constraints, i.e. the contents of each test tube is distributed to all test tubes according to specific output and input filters again, whereas the rest remains in the test tube. These ideas have already be formalized for the splicing operation in [4] and for the operations of cutting and recombination in [9].

Definition 3. A *test tube system with output and input filters* (a *TTSOI* for short) σ is a sextuple (B, n, A, ρ, O, I), where

1. B is a set of *objects;*
2. n, $n \geq 1$, is the number of test tubes in σ;
3. $A = (A_1, ..., A_n)$, where A_i is a set of *axioms,* which are elements from B, $1 \leq i \leq n$;
4. ρ is a sequence $(\rho_1, ..., \rho_n)$ of sets of *test tube operations,* where ρ_i contains specific operations for the test tube T_i, $1 \leq i \leq n$;
5. $O = (O_1, ..., O_n)$, where $O_i \subseteq B$ is the *output filter* for the test tube T_i, $1 \leq i \leq n$;
6. $I = (I_1, ..., I_n)$, where $I_i \subseteq B$ is the *input filter* for the test tube T_i, $1 \leq i \leq n$.

In order to indicate the number n of test tubes, we also call σ a TTSOI_n.

The computations in the system σ 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 T_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 and input filters. From $\rho_i^*(L_i)$ only the part $(\rho_i^*(L_i) \cap O_i) \cap I_j$ that passes the output filter O_i as well as the input filter I_j is distributed to the test tube T_j, $1 \leq j \leq n$, whereas the rest

$$\rho_i^*(L_i) \setminus \left(\bigcup_{1 \leq j \leq n} ((\rho_i^*(L_i) \cap O_i) \cap I_j) \right)$$

remains in T_i. The final result of the computations in σ consists of all objects from B that can be extracted from the *final test tube* T_1 via the ouput filter O_1.

More formally, an *instantaneous description* (ID for short) of the system σ is an n-tuple $(L_1, ..., L_n)$ with $L_i \subseteq B$, $1 \leq i \leq n$, where L_i describes the contents of test tube T_i at the beginning of a derivation step. The initial ID is $(A_1, ..., A_n)$, i.e. at time $t = 0$ the test tubes T_i contain the axioms A_i. Let $(L_1(t), ..., L_n(t))$

denote the ID at time t; then one derivation step with the system σ yields the ID $(L_1(t+1), ..., L_n(t+1))$, where

$$L_i(t+1) = \left(\bigcup_{1 \leq j \leq n} (\rho_j^*(L_j(t) \cap O_j) \cap I_i)\right) \cup \left(\rho_i^*(L_i(t)) \setminus \left(\bigcup_{1 \leq j \leq n} ((\rho_i^*(L_i(t)) \cap O_i) \cap I_j)\right)\right).$$

We also write $(L_1(t), ..., L_n(t)) \vdash_\sigma (L_1(t+1), ..., L_n(t+1))$. The language generated by the system σ, $L(\sigma)$, then is defined by $L(\sigma) = \bigcup_{t=0}^{\infty} (L_1(t) \cap O_1)$. Moreover, we say that σ is of type (F_1, F_2, F_3, F_4), if $A_i \in F_1$, $\rho_i \in F_2$, $O_i \in F_3$, and $I_i \in F_4$ for all i with $1 \leq i \leq n$. □

Definition 4. A *test tube system with input filters* (a *TTSI* for short) σ of type (F_1, F_2, F_4) is a quintuple (B, n, A, ρ, I), where $(B, n, A, \rho, (B, ..., B), I)$ is the corresponding TTSOI of type $(F_1, F_2, \{B\}, F_4)$. A *test tube system with output filters* (a *TTSO* for short) σ of type (F_1, F_2, F_3) is a quintuple (B, n, A, ρ, O), where $(B, n, A, \rho, O, (B, ..., B))$ is the corresponding TTSOI of type $(F_1, F_2, F_3, \{B\})$. In order to indicate the number n of test tubes, we also call σ a TTSI_n and a TTSO_n, respectively. □

We should like to mention that in general the TTSOI $(B, n, A, \rho, (B, ..., B), I)$ corresponding with a TTSI (B, n, A, ρ, I) of type (F_1, F_2, F_4) need not be a TTSOI of type (F_1, F_2, F_4, F_4), because B need not be an element of F_4.

Remark 1. The reader should observe that we are not dealing with multisets in this paper; hence we assume that every object in any test tube is available in an unbounded number. Moreover, in the phase of redistribution every object x from the test tube T_i that passes the output filter O_i is distributed (in an unbounded number) to each test tube T_j the input filter of which allows x to pass. In some sense this corresponds to an intermediate step which in practice is called *amplification*, e.g. in test tubes working with DNA strands (and the operation of splicing) copies can be made by applying the polymerase chain reaction (see [3]). Moreover, it would be a more practical assumption that instead of $\rho_i^*(L_i)$ any arbitrary (finite) subset of $\rho_i^*(L_i)$ could evolve in the test tube T_i during a computation period. Then only this subset would be distributed to all test tubes according to the input filters. In fact, in most cases this would still allow us to generate all desired objects, although it would never be clear, when these objets would evolve. In a practical implementation the number and the size of objects that can be generated also depends on the amount of original material of axioms we take at the beginning. Moreover, if parts of (the subset of) $\rho_i^*(L_i)$ are to be redistributed over different test tubes it is only necessary to assume that any allowed distribution of the whole material will possibly happen; in practical implementations of test tube systems the intermediate amplification (see [2], [3], [12]) of the material may already guarantee that enough material is distributed to all the possible test tubes. □

A minimal requirement on the feasability of the input filters I_i and the output filters O_j is their recursiveness, i.e. we demand that it is decidable whether an object from B can pass a filter or not.

The following example shows how under these constraints every recursive language can be generated by a large class of TTSO_1:

Example 1. Let $L \subseteq V^+$ be an arbitrary recursive language and let σ_L be the TTSO_1 $\sigma_L = (B, 1, (A), (\rho), (L))$ such that $\rho^*(A) \supseteq L$. Then we obtain $(A) \vdash_{\sigma_L} (\rho^*(A) \cap L) = (L)$ and therefore $L(\sigma) = (A \cap L) \cup (\rho^*(A) \cap L) = L$.

Hence, for any family of languages F_3 with $F_3 \subseteq REC$, a language $L \subseteq F_3$ can be generated by a TTSO_1 of type (F_1, F_2, F_3) if F_1 contains a set A such that $\rho^*(A) \supseteq L$ for some $\rho \in F_2$. □

Definition 5. A *CR-TTSOI* σ is a TTSOI $(O(V, M), n, A, \rho, O, I)$, where $\rho = (\rho_1, ..., \rho_n)$, $\rho_i = C_i \cup R_i$, $1 \leq i \leq n$, and $\sigma_i = (V, M, C_i, R_i)$ is a CR-scheme. In order to emphasize that σ is a CR-TTSOI, we shall also write (C_i, R_i) for ρ_i instead of $C_i \cup R_i$. An H-TTSOI σ is a TTSOI (V^+, n, A, ρ, O, I), where $\sigma_i = (V, \rho_i)$, $1 \leq i \leq n$, is an H-scheme. A G-TTSOI σ is a TTSOI $(W(V_N, V_T), n, A, \rho, O, I)$, where V_N and V_T are disjoint alphabets, $W(V_N, V_T)$ denotes $(V_N \cup V_T)^+$, and $\sigma_i = (V_N, V_T, \rho_i)$, $1 \leq i \leq n$, is a grammar scheme; if every grammar scheme σ_i, $1 \leq i \leq n$, is context-free (regular), then also σ is called context-free (regular). □

Remark 2. The notation $W(V_N, V_T)$ in a G-TTSOI σ, $\sigma = (W(V_N, V_T), n, A, \rho, O, I)$, allows us to distinguish the non-terminal symbols in V_N and the terminal symbols in V_T; σ is considered to work "correctly" only if $L(\sigma) \subseteq V_T^+$. In a similar manner for a CR-TTSOI σ, $\sigma = (O(V, M), n, A, \rho, O, I)$, we demand $L(\sigma) \subseteq V^+$. □

We now exhibit an example of a regular G-TTSO_7 of type (FIN, FIN, REG^+) which generates a non-context-free language:

Example 2. Let $\sigma = (W(V_N, V_T), 7, A, \rho, O)$ be the G-TTSO_7 with

$V_N = \{X, Y\}$, $V_T = \{a, b\}$,

$A = (\emptyset, \{XX\}, \{XX\}, \{XX\}, \{XX\}, \emptyset, \emptyset)$,

$\rho = (\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, \rho_7)$, $O = (O_1, O_2, O_3, O_4, O_5, O_6, O_7)$.

$\rho_1 = \emptyset$, $\rho_2 = \{(X, a), (Y, a)\}$, $\rho_3 = \{(X, b), (Y, b)\}$,

$\rho_4 = \{(X, aY)\}$, $\rho_5 = \{(X, bY)\}$, $\rho_6 = \{(Y, aX)\}$, $\rho_7 = \{(Y, bX)\}$,

$O_1 = O_2 = O_3 = \{a, b\}^+$, $O_4 = O_5 = \{a, b, Y\}^+$, $O_6 = O_7 = \{a, b, X\}^+$.

The generation of the words ww in this G-TTSO briefly can be described in the following way:

From arbitrary words of the forms $uXuX$, $u \in \{a, b\}^*$, and $vYvY$, $v \in \{a, b\}^+$, respectively, by the productions in ρ_2 we obtain $uaua$ and $vava$, respectively, whereas by the productions in ρ_3 we obtain $ubub$ and $vbvb$, respectively, i.e. we obtain terminal words from $\{a, b\}^+$, which then can be extracted from T_1 as terminal results of the computations in σ. By the corresponding productions in ρ_4, ρ_5, ρ_6, ρ_7, the length of the current strings is prolongued by one more symbol in a synchronized way, because only the strings of the forms $uaYuaY$, $ubYubY$, $vaXvaX$, and $vbXvbX$, respectively, can pass the corresponding output filters O_i of the test tubes T_i, $4 \leq i \leq 7$. These observations show that $L(\sigma) = \{ww \mid w \in \{a, b\}^+\}$. □

For CR-TTSOI the following types of filters are suitable:

Definition 6. 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. □

The proof of the following result is obvious from the definitions and therefore omitted:

Lemma 1. The union and the intersection of two $(V,M)_i$-filters again is a $(V,M)_i$-filter, $i \in \{1,2\}$. Moreover, $O(V,M)$ is a $(V,M)_2$-filter, but not a $(V,M)_1$-filter.

The distribution of the contents of a test tube over all test tubes of the system not only gives rise to theoretical problems (for obtaining filters of a complexity as low as possible) but also to practical problems ("waste" of the material that is put into test tubes where on one hand it cannot be processed or used any more and on the other hand it nonetheless has to remain forever). Hence, a more natural and realistic scenario is to assume that the contents of the test tubes is only distributed to selected test tubes that are prescribed from the beginning. In fact, most of the test tube systems to be found in literature work in such a manner, i.e. *programs* how to redistribute the contents of test tubes are described (see [1], [3], [12]).

A formalization of these ideas discussed above leads to the following definition:

Definition 7. A *test tube system with prescribed output/input relations* (a *TTSPOI* for short) σ is a quintuple (B,n,A,ρ,D), where

1. B is a set of *objects;*
2. n, $n \geq 1$, is the number of test tubes in σ;
3. $A = (A_1,...,A_n)$ is a sequence of sets of *axioms,* where $A_i \subseteq B$, $1 \leq i \leq n$;
4. ρ is a sequence $(\rho_1,...,\rho_n)$ of sets of *test tube operations,* where ρ_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 σ 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 σ is a TTSPOI$_n$.

The computations in the system σ 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 ρ_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 σ 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 $(1,F,j) \in D$).

More formally, an *instantaneous description* (ID for short) of the system σ is an n-tuple $(L_1, ..., L_n)$ with $L_i \subseteq B$, $1 \leq i \leq n$, where L_i describes the contents of test tube T_i at the beginning of a derivation step. The initial ID is $(A_1, ..., A_n)$, i.e. at time $t = 0$ test tubes T_i contain the axioms A_i. Let $(L_1(t), ..., L_n(t))$ denote the ID at time t; then one derivation step with the system σ yields the ID $(L_1(t+1), ..., L_n(t+1))$, where

$$L_i(t+1) = \left(\bigcup_{(j,F,i) \in D} (\rho_j^*(L_j(t)) \cap F) \right) \cup \left(\rho_i^*(L_i(t)) \setminus \bigcup_{(i,F,j) \in D} (\rho_i^*(L_i(t)) \cap F) \right).$$

We also write $(L_1(t), ..., L_n(t)) \vdash_\sigma (L_1(t+1), ..., L_n(t+1))$. The language generated by the system σ, $L(\sigma)$, is defined by $L(\sigma) = \bigcup_{t=0}^{\infty} L_1(t)$. Moreover, we say that σ 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 $1 \leq i \leq n$, $1 \leq j \leq n$. □

Definition 8. A *CR-TTSPOI* σ 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. An H-TTSPOI σ is a TTSPOI (V^+, n, A, ρ, D) where $\sigma_i = (V, \rho_i)$, $1 \leq i \leq n$, is an H-scheme. A G-TTSPOI σ is a TTSPOI $(W(V_N, V_T), n, A, \rho, D)$, where $\sigma_i = (V_N, V_T, \rho_i)$, $1 \leq i \leq n$, is a grammar scheme; if every grammar scheme σ_i, $1 \leq i \leq n$, is context-free (regular), then also σ is called context-free (regular). □

Remark 3. As already stated in Remark 2 for CR-TTSOI and G-TTSOI, respectively, also for a CR-TTSPOI σ, $\sigma = (O(V,M), n, A, \rho, D)$, we demand $L(\sigma) \subseteq V^+$ and for a G-TTSPOI σ, $\sigma = (W(V_N, V_T), n, A, \rho, D)$, we demand $L(\sigma) \subseteq V_T^+$.

The following results were established in [9]:

Proposition 1. For every $L \in ENUM$ we can construct a CR-TTSI of type (FIN, FIN, CRF_1) which generates L.

Proposition 2. For every $L \in ENUM$ we can construct a CR-TTSPOI$_4$ of type (FIN, FIN, CRF_2) which generates L.

For the CR TTSI in Proposition 1 it is an open question whether the number of test tubes needed for generating arbitrary recursively enumerable languages can be bounded or not.

The following result was proved in [4]:

Proposition 3. For every $L \in ENUM\,(V_T)$ we can construct an H-TTSI$_{8+card(V_T)}$ of type (FIN, FIN, REG^+) which generates L.

3 Results

In the first part of this section we elaborate some general relations between the different models of test tube systems we introduced in the previous section; these results even hold true for arbitrary objects and for arbitrary operations used in the test tubes. In the second part of this section we shall prove some specific results for test tube systems working on strings, e.g. we shall show how every recursively enumerable string language can be generated by test tube systems using context-free productions and a very restricted form of regular filters only.

3.1 General results for test tube systems

In this subsection we show some general results for the different models of test tube systems introduced in the previous section; these results neither depend on the operations used in the test tubes nor on the objects we consider. Moreover, we give some applications of these general results for CR test tube systems.

For every TTSOI we can easily construct an equivalent TTSPOI generating the same language:

Lemma 2. Let $\sigma = (B, n, A, \rho, O, I)$ be a TTSOI$_n$ of type (F_1, F_2, F_3, F_4) and let F_0 be a set containing $L(\sigma)$. Then the TTSPOI$_{n+1}$ $\sigma' = (B, n+1, (\emptyset, \rho_1, ..., \rho_n), (\emptyset, A_1, ..., A_n), D)$ with

$$D = \{(i+1, O_i \cap I_j, j+1) \mid 1 \leq i, j \leq n\} \cup \{(2, F_0 \cap O_1, 1)\}$$

generates the same language as σ, i.e. $L(\sigma') = L(\sigma)$. Let F_1 and F_2 contain the empty set and denote $\sqcap(F_3, F_4) = \{X \cap Y \mid X \in F_3 \wedge Y \in F_4\}$; then σ' is a TTSPOI$_{n+1}$ of type (F_1, F_2, F_5) for every family F_5 with $F_5 \supseteq \{F_0 \cap O_1\} \cup \sqcap(F_3, F_4)$.

Proof. The components (test tubes) T'_{i+1}, $1 \leq i \leq n$, in σ' work in the same way as the corresponding test tubes T_i in σ, because by definition they contain the same rules, i.e. $\rho'_{i+1} = \rho_i$. We also start with the desired axioms A_j in each test tube T'_{j+1}, $1 \leq j \leq n$. The output/input relations $(i+1, O_i \cap I_j, j+1)$, $1 \leq i \leq n$, $1 \leq j \leq n$, guarantee that the test tubes T'_{i+1} in σ' are distributed in the same way as the test tubes T_i in σ after each computation step. The test tube T'_1 is only needed to extract the final objects in σ' in a similar way as by extracting these strings from T_1 in σ. In sum we obtain

$$\begin{aligned} (A_1, ..., A_n) &\vdash_\sigma^* (L_1(t), ..., L_n(t)) \\ &\vdash_\sigma (L_1(t+1), ..., L_n(t+1)) \end{aligned}$$

if and only if

$$\begin{aligned}(\emptyset, A_1, ..., A_n) \quad &\vdash_{\sigma'}^{*} \quad (L_0(t), L_1(t), ..., L_n(t)) \\ &\vdash_{\sigma'} \quad (L_0(t) \cup (L_1(t) \cap (F_0 \cap O_1)), L_1(t+1), ..., L_n(t+1)),\end{aligned}$$

which immediately yields

$$\begin{aligned}L(\sigma') = \; &\bigcup_{t=0}^{\infty} L_0(t) = \emptyset \cup \bigcup_{t=0}^{\infty} (L_1(t) \cap (F_0 \cap O_1)) = \\ &(\bigcup_{t=0}^{\infty} L_1(t) \cap O_1) \cap F_0 = L(\sigma) \cap F_0 = L(\sigma).\end{aligned}$$

□

Corollary 1. For every $L \in ENUM(V_T)$ we can construct a CR-TTSPOI of type (FIN, FIN, CRF_1) which generates L.

Proof. From Proposition 1 we know that for L we can construct a CR-TTSOI $\sigma = (O(V,M), n, \rho, (O(V,M), ..., O(V,M)), I)$ of type $(FIN, FIN, \{O(V,M)\}, CRF_1)$ with $L(\sigma) = L$. Now take $F_0 = V^+$ (observe that $V^+ \in CRF_1$); obviously, for any $F \in CRF_1$ we have $F \cap O(V,M) = F$ and therefore $\{F_0 \cap O_1\} \cup (\sqcap(\{O(V,M)\}, CRF_1)) \subset CRF_1$; hence, we can apply Lemma 2. □

Because of Lemma 1, the following result, for instance, holds true for CR-TTSPOI of type (F_1, F_2, CRF_i), $i \in \{1,2\}$:

Lemma 3. Let $\sigma = (B, n, A, \rho, D)$ be a TTSPOI_n of type (F_1, F_2, F_3). If the family of filters F_3 contains the empty set and is closed under union, then we can construct an equivalent TTSPOI_n $\sigma' = (B, n, A, \rho, D')$ of the same type (F_1, F_2, F_3) such that for any two test tubes there is exactly one output/input relation (by a filter from F_3), and moreover, the derivation relations $\vdash_\sigma$ and $\vdash_{\sigma'}$ are identical.

Proof. The result is obvious by defining

$$D' = \bigcup_{1 \le i,j \le n} \left\{ (i, F_{i,j}, j) \mid F_{i,j} = \bigcup_{(i,F,j) \in D} F \right\}.$$

Observe that $F_{i,j}$ is empty if in D no output/input relation between T_i and T_j exists. □

Remark 4. If we want to use the filters $F_{i,j,1}, ..., F_{i,j,k}$ only instead of the union filter $\bigcup_{m=1}^{k} F_{i,j,m}$ between the test tubes T_i and T_j, but still do not want to have more than one connection between two test tubes, we have to add k additional test tubes $T_{i,j,1}, ..., T_{i,j,k}$, which with (time) delay one contain the same strings as T_i, and then from $T_{i,j,m}$ by the filter $F_{i,j,m}$ distribute this filtered part of T_i to T_j, i.e. $\rho_{i,j,m} = \emptyset$, and instead of the output/input relations $(i, F_{i,j,m}, j)$, $1 \le m \le k$, we have the relations $(i, F_{i,j,m}, (i,j,m))$ and $((i,j,m), F_{i,j,m}, j)$ for all m with $1 \le m \le k$. □

As an immediate consequence of these considerations, $(V,M)_i$-filters, $i \in \{1,2\}$, which by definition are finite unions of simple $(V,M)_i$-filters, in the same way can be split up into their components. Hence, Corollary 1 now can be sharpened to the following result which shows that in CR-TTSPOI we only need simple $(V,M)_i$-filters in order to obtain full generative power:

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

Under specific constraints, a TTSPOI$_n$ can even be simulated by a TTSOI$_n$:

Lemma 4. Let $\sigma = (B,n,A,\rho,D)$ be a TTSPOI$_n$ of type (F_1,F_2,F_3) such that

1. the family of filters F_3 contains the empty set and is closed under union;
2. for any two test tubes T_i and T_k with $1 \leq i \leq n$, $1 \leq k \leq n$, there is exactly one output/input relation $(i,F_{i,k},k)$ with $F_{i,k} \in F_3$;
3. for all $(i,F_{i,j},j)$ and $(i,F_{i,k},k)$ in D with $j \neq k$ we have $F_{i,j} \cap F_{i,k} = \emptyset$;
4. for all k with $1 \leq k \leq n$ we have $F_{1,k} = \emptyset$;

then we can construct an equivalent TTSOI$_n$ $\sigma' = (B,n,A,\rho,O,I)$ of the type (F_1,F_2,F_3,F_3) such that $L(\sigma') = L(\sigma)$.

Proof. The desired result is obvious by defining $O_i = \bigcup_{1 \leq j \leq n} F_{i,j}$ and $I_i = \bigcup_{1 \leq j \leq n} F_{j,i}$ for $2 \leq i \leq n$ as well as $O_1 = I_1 = \bigcup_{1 \leq j \leq n} F_{j,1}$. □

3.2 Some specific results for test tube systems

In this subsection we shall show how any recursively enumerable language can be generated by test tube systems with context-free productions and regular filters:

Theorem 1. For every $L \in ENUM(V_T)$ we can construct a context-free G-TTSPOI$_3$ of type (FIN, FIN, REG) which generates L.

Proof. Without loss of generality we may assume that L is given by a grammar G in Geffert normal form (see [10]), i.e. $G = (\{S,A,B,C\}, V_T, P_{cf} \cup \{ABC \to \lambda\}, S)$, where P_{cf} contains only context-free productions of the form (S,w). Now we define

$$
\begin{aligned}
\sigma &= (W(V_N', V_T), 3, (\emptyset, \{S\}, \emptyset), (\emptyset, \rho_2, \rho_3), D), \\
V_N' &= \{S,A,B,C,A',B',C'\}, \\
V &= \{S,A,B,C\} \cup V_T, \\
\rho_2 &= P_{cf} \cup \{A \to A', B \to B', C \to C'\}, \\
\rho_3 &= \{A' \to \lambda, B' \to \lambda, C' \to \lambda\}, \\
D &= \{(2, V_T^+, 1), (2, V^* \{A'B'C'\} V^*, 3), (3, V_T^+, 1), (3, V^+ \setminus V_T^+, 2)\} \cup \\
&\quad \{(1,\emptyset,1), (1,\emptyset,2), (1,\emptyset,3), (2,\emptyset,2), (3,\emptyset,3)\}.
\end{aligned}
$$

Whenever an application of the only non-context-free rule $ABC \to \lambda$ has to be simulated in σ, we have to apply the productions $A \to A', B \to B', C \to C'$ in T_2 in

such a manner that the resulting word can pass the filter $V^* \{A'B'C'\} V^*$, which checks the context condition; the final *execution* of the simulation is carried out by the productions $A' \to \lambda, B' \to \lambda, C' \to \lambda$ in T_3. Only terminal words are passed from T_2 and T_3 to T_1. Hence we conclude $L(\sigma) = L$. □

As the construction elaborated in the preceding proof fulfills the necessary assumptions, we immediately can apply Lemma 4, which shows that we can construct a context-free G-TTSOI$_3$ of type (FIN, FIN, REG) which generates L; yet we can even get more, i.e. we only need a context-free G-TTSI$_3$ or a G-TTSO$_3$ of type (FIN, FIN, REG) for generating L :

Corollary 3. For every $L \in ENUM(V_T)$ we can construct a context-free G-TTSI$_3$ which generates L as well as a context-free G-TTSO$_3$ of type (FIN, FIN, REG) which generates L.

Proof. In a similar way as in the proof of Theorem 1 we define the context-free G-TTSI$_3$

$$\sigma_I = (W(V'_N, V_T), 3, (\emptyset, \{S\}, \emptyset), (\emptyset, \rho_2, \rho_3), I)$$

and the context-free G-TTSO$_3$

$$\sigma_O = (W(V'_N, V_T), 3, (\emptyset, \{S\}, \emptyset), (\emptyset, \rho_2, \rho_3), O)$$

where V', ρ_2, ρ_3 are defined as in the proof of Theorem 1 as well as

$$I_1 = V_T^+, I_2 = V^+, I_3 = V^* \{A'B'C'\} V^*, \text{ and}$$
$$O_1 = V_T^+, O_2 = V_T^+ \cup V^* \{A'B'C'\} V^*, O_3 = V^+.$$

It is easy to verify that $L(\sigma_I) = L(\sigma_O) = L(\sigma) = L$. □

The results in Theorem 1 and Corollary 3 are optimal in the sense that a context-free G-TTSPOI$_2$ of type (FIN, FIN, REG) can only generate context-free languages:

Theorem 2. For every context-free G-TTSPOI$_2$ σ of type (FIN, FIN, REG), $L(\sigma) \in CF$.

Proof. Let $\sigma = (W(V_N, V_T), 2, (A_1, A_2), (\rho_1, \rho_2), D)$ be a context-free G-TTSPOI of type (FIN, FIN, REG), i.e. ρ_1 and ρ_2 contain only context-free productions, and the filters in D are regular. The elements of the first test tube must be terminal words, hence no context-free productions can be applied any more to these words, neither in the first test tube nor, after distribution according to an output/input relation $(1, F_{1,2}, 2)$, in the second test tube, hence we can assume $\rho_1 = \emptyset$ as well as $D = \{(2, F_{2,1}, 1), (2, F_{2,2}, 2), (1, \emptyset, 1), (1, \emptyset, 2)\}$, where $F_{2,1}$ and $F_{2,2}$ are regular languages. $F_{2,1}$ only has the task to extract terminal strings from the contents of the second test tube, i.e. the relation $(2, F_{2,1}, 1)$ only works as a final intersection with the regular set $F_{2,1}$. During the first derivation step, in T_2 from A_2 we obtain

$\rho_2^*(A_2)$. As $\rho_2^*(A_2) \cap F_{2,2} \subseteq \rho_2^*(A_2)$ and $\rho_2^*(\rho_2^*(A_2)) = \rho_2^*(A_2)$, in further derivation steps no additional strings can evolve in T_2. Hence, as the family of context-free languages is closed under union as well as under intersection with regular sets, we obtain $\rho_2^*(A_2) \in CF$, $L(\sigma) = \rho_2^*(A_2) \cap F_{2,1}$, and therefore $L(\sigma) \in CF$. □

Remark 5. In a similar way as above it is easy to show that for every regular G-TTSPOI$_2$ σ of type (FIN, FIN, REG), $L(\sigma) \in REG$. Obviously, for any context-free G-TTSPOI$_1$ σ of type (FIN, FIN, REG) with $\sigma = (W(V_N, V_T), (A_1), (\rho_1), \{(1, F_{1,1}, 1)\})$ we have $L(\sigma) = A_1$, because the words in A_1 must not contain non-terminal symbols; hence, only finite languages can be generated. On the other hand, the language generated by a regular respectively by a context-free G-TTSO$_1$ with $\sigma = (W(V_N, V_T), (A_1), (\rho_1), (F_1))$ is $\rho_1^*(A_1) \cap F_1$, i.e. as $REG(V_T)$ and $CF(V_T)$ are closed under union as well as under intersection with regular sets, such G-TTSO$_1$ exactly characterize $REG(V_T)$ and $CF(V_T)$, respectively. □

Remark 6. The existence of a universal grammar scheme $\gamma_U(V_T)$ for V_T and the results shown above also imply the existence of a universal context-free G-TTSI$_3$ $\sigma_U(V_T)$ for V_T, $\sigma_U(V_T) = (W(V'_N, V_T), 3, (\emptyset, \emptyset, \emptyset), (\emptyset, \rho_2, \rho_3), I)$, where ρ_2 and ρ_3 contain context-free productions and the filters in I are regular, such that for every $L \in ENUM(V_T)$ the context-free G-TTSI$_3$ σ_L, $\sigma_L = (W(V'_N, V_T), 3, (\emptyset, \{A_L\}, \emptyset), (\emptyset, \rho_2, \rho_3), I)$, generates L, where A_L denotes the initial word used for $\gamma_U(V_T)$ to obtain a grammar generating L. □

4 Summary and Future Research

In this paper we introduced various general models of test tube systems. We investigated several general relations between different kinds of these models and also showed some specific results, e.g. how to generate any arbitrary recursively enumerable language by a test tube system with context-free productions and a restricted type of regular filters.

Special practical variants have already been described in literature for solving very specific problems in the area of DNA computing and the construction of molecular computers based on test tubes has been considered by using different operations on the test tubes (e.g. see [1], [2], [3], [12]). In [4], test tube systems based on the splicing operation were shown to allow the construction of universal mechanisms; a similar result was shown for test tube systems based on the operations of cutting and recombination in [9]. Various other types of test tube systems based on context-free productions can also be shown to be computationally universal as we have exhibited in the previous section.

The general results proved in the first part of the preceding section also hold true for test tube systems working on other objects than strings, e.g. for circular strings, graphs, and arrays. Hence, there is a wide field of interesting problems to be considered in the future.

Acknowledgements

We gratefully appreciate fruitful discussions with Erzsébet Csuhaj-Varjú and Gheorghe Păun on some of the topics considered in this paper.

References

[1] L. M. Adleman, Molecular computation of solutions to combinatorial problems, *Science*, 226 (Nov. 1994), 1021 – 1024.

[2] L. M. Adleman, On constructing a molecular computer, manuscript, January 1995.

[3] D. Boneh, C. Dunworth, R.J. Lipton, J. Sgall, On the computational Power of DNA, *to appear.*

[4] E. Csuhaj-Varjú, L. Kari, and Gh. Păun, Test tube distributed systems based on splicing, *Computers and Artificial Intelligence*, Vol. 15 (2) (1996), 211-232

[5] E. Csuhaj-Varjú and A. Salomaa, Networks of parallel language processors, submitted.

[6] E. Csuhaj-Varjú, Networks of language processors. A survey. In: *Lenguajes Naturales Y Lenguajes Formales XII*, C. Martin-Vide, ed., PPU, Barcelona, 1996, pp. 169-189.

[7] R. Freund, L. Kari, and Gh. Păun, DNA computing based on splicing: The existence of universal computers, Techn. Report 185-2/FR-2/95, TU Wien, 1995.

[8] R. Freund and F. Wachtler, Universal systems with operations related to splicing, *Computers and Artificial Intelligence*, Vol. 15 (4) (1996), 273-294.

[9] R. Freund, E. Csuhaj-Varjú, and F. Wachtler, Test tube systems with cutting/recombination operations, Proceedings PSB'97, 1997.

[10] V. Geffert, Context-free-like forms for the phrase-structure grammars, *Proceedings MFCS'88*, Lecture Notes in Computer Science, Vol. 324, Springer-Verlag, Berlin (1988), 309 – 317.

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

[12] R. J. Lipton, Speeding up computations via molecular biology, *manuscript*, December 1994.

[13] Gh. Păun, Regular extended H systems are computationally universal, *J. of Automata, Languages and Combinatorics*, Vol. 1, Nr. 1 (1996), 27 – 37.

[14] P. W. K. Rothemund, A DNA and restriction enzyme implementation of Turing machines, *manuscript*, 1995.

[15] R. Siromoney, K. G. Subramanian, and V. R. Dare, Circular DNA and splicing systems, Lecture Notes in Computer Science, Vol. 654, Springer-Verlag, Berlin (1992), 260 – 273.

[16] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.