Malaysian Journal of Fundamental & Applied Sciences

available online at http://mjfas.ibnusina.utm.my

Splicing Systems over Permutation Groups of Length Two

N.Z.A. Hamzah[1*], N.A. Mohd Sebry[1], W.H. Fong[2], N.H. Sarmin[1, 2] and S. Turaev[3]

[1] *Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, Malaysia*
[2] *Ibnu Sina Institute for Fundamental Science Studies, Universiti Teknologi Malaysia, Malaysia*
[3] *Faculty of Computer Science and Informational Technology, Universiti Putra Malaysia, Malaysia.*

Received 20 October 2011, Revised 21 January 2012, Accepted 5 February 2012, Available online 15 February 2012

ABSTRACT

The first theoretical model of DNA computing, called a *splicing system*, for the study of the generative power of deoxyribonucleic acid (DNA) in the presence of restriction enzymes and ligases was introduced by Head in 1987. Splicing systems model the recombinant behavior of double-stranded DNA (dsDNA) and the enzymes which perform operation of cutting and pasting on dsDNA. Splicing systems with finite sets of axioms and rules generate only regular languages when no additional control is assumed. With several restrictions to splicing rules, the generative power increase up to recursively enumerable languages. Algebraic structures can also be used in order to control the splicing systems. In the literature, splicing systems with additive and multiplicative valences have been investigated, and it has been shown that the family of languages generated by valence splicing systems is strictly included in the family of context-sensitive languages. This motivates the study of splicing systems over permutation groups. In this paper, we define splicing systems over permutation groups and investigate the generative power of the languages produced.

| Splicing System | Generative power | Regular Languages | Recursively Enumerable Languages | Permutation groups |

http://dx.doi.org/10.11113/mjfas.v8n2.127

1. INTRODUCTION

The study of splicing system was first introduced by Head in 1987 to model the generative power of deoxyribonucleic acid (DNA) in the presence of restriction enzymes and ligases. Head introduced splicing system as a mathematical model of the recombinant behavior of double-stranded DNA (dsDNA) and the enzymes which perform operation of cutting and pasting on dsDNA. Splicing system produces a language called splicing language [1]. In [2], all splicing languages with finite sets of axioms and rules have been proved to be regular. But not all regular languages are splicing languages [3].

Groups have been used as a control mechanism in splicing systems since splicing system with finite sets of axioms and rules generate only regular languages. Thus, in order to increase the generative power of splicing systems, several restrictions are imposed on the splicing systems. One kind of such restrictions is to control the splicing operation by valences associated with the strings [4]. In this restriction, the given valences are the elements of monoid or group that are associated with the axioms. The strings are accepted if the valences of the strings are equal to the identity element.

In this paper, splicing system over permutation group is defined and the languages produced by permutation group of length two are determined in the study of splicing languages with valences.

*Corresponding author at:
E-mail address: nz_akmar@yahoo.com (Nur Zatul Akmar Bt Hamzah)

2. PRELIMINARIES

Some formal definitions related to this research will be presented in this section. The main definition which is splicing system will first be defined.

Definition 1 [1] (Splicing System)
A **splicing system** $S = (V, I, B, C)$ consists of a finite alphabet V, a finite set I of initial strings in V^*, and finite sets B and C of triples (c, x, d) with c, x and d in V^*, where V^* is the set of all strings over an alphabet V. For each such triple the string cxd is called a site and the string x is called a crossing. A language L is a splicing language if there exists a splicing system S for which $L = L(S)$.

When some restriction enzymes and ligases are present in a test tube, they do not stop acting after one cut and paste operation, but they act respectively on DNA [5].

A new model of computation based on the splicing operation called an H system has been investigated by some researchers [4, 5, 6]. The definitions of H scheme, H system and extended H system are shown respectively in the following.

Definition 2 [6] (H Scheme)
An **H scheme** is a pair $\sigma = (V, R)$, where V is an alphabet and $R \subseteq V^*\#V^*\$V^*\#V^*$ is a set of splicing rules over V, where V^* is the set of all strings over an alphabet V. The symbols $ and # are not in V.

Definition 3 [6] (H System)
An **H System** is a triple $\gamma = (V, A, R)$ with respect to a pair (σ, A), where $\sigma = (V, R)$ is an H scheme and $A \subseteq V^*$ is the set of axioms and $r = u_1\#u_2\$u_3\#u_4 \in R$ is the set of splicing rules, for some $x_1, x_2, y_1, y_2 \in V^*$, then $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, $z = x_1u_1u_4y_2$, $w = y_1u_3u_2x_2$.

Therefore, $\sigma(A) = \{z,\ w \in V^* \mid (x, y) \vdash_r (z, w)$ for some $x, y \in A$, $r \in R\}$, and $\sigma^*(A) = \bigcup_{i \geq 0} \sigma^i(A)$, where $\sigma^0(A) = A$, and $\sigma^{i+1}(A) = \sigma^i(A) \cup \sigma(\sigma^i(A))$, $i \geq 0$. The language generated by an H system γ is defined by $L(\gamma) = \sigma^*(A)$.

Definition 4 [6] (Extended H System)
An **extended H system** is a construct of $\gamma = (V, T, A, R)$ where V is an alphabet, $T \subseteq V$ is the terminal alphabet, $A \subseteq V^*$ is the set of axioms, and $R \subseteq V^*\#V^*\$V^*\#V^*$ is the set of splicing rules.

When $T = V$, the system is said to be a non-extended H system. The pair $\sigma = (V, R)$ is the underlying H scheme of γ. The language generated by γ is defined by $L(\gamma) = \sigma^*(A) \cap T^*$.

An example of extended H system is shown in the following.

Example 1 [6] Extended H system
Consider an extended H system $\gamma = (\{a, b, c\}, \{a\}, \{a^n b, ca^m\}, \{a\#b\ \$\ c\#a\})$, where n, m are two given positive integers. The only splicing rule that can be applied to the two existing axioms, $a^n b$ and ca^m, is $a\#b\ \$\ c\#a$. The operation of this splicing is $(a^n \mid b,\ c \mid a^m) \vdash (a^{n+m},\ cb)$. No other new strings can be obtained in this splicing operations of extended H system using the rule $a\#b\ \$\ c\#a$.

Then for $A = \{a^n b, ca^m\}$, the computation of extended splicing system is shown in the following:

$$\sigma^0(A) = \{a^n b, ca^m\},$$
$$\sigma^1(A) = \sigma^0(A) \cup \{a^{n+m}, cb\},$$
$$\sigma^i(A) = \sigma^1(A),\ i \geq 2.$$

Therefore, $\sigma^*(A) = \sigma^0(A) \cup \sigma^1(A) = \{a^n b,\ ca^m,\ cb,\ a^{n+m}\}$ and $L(\gamma) = \{a^{n+m}\}$.

Next, permutation groups can be used as valences in extended H systems. The definition of permutation group is stated in the following.

Definition 5 [7] (Permutation Group)
Let A be a finite set {1, 2, 3, 4, .. , n}. The group of all permutations of A or **permutation group** A is the symmetric group on n letters and is denoted by S_n.

Then, the definition of Chomsky grammar that is used to determine the generative power of valences in splicing systems is also presented.

Definition 6 [6] (Chomsky Grammar)
A **Chomsky grammar** is a quadruple $G = (N, T, S, P)$ where:
N is a finite set of nonterminals,
T is a finite set of terminals,
$S \in N$ is the start symbol (the axiom),
P is a finite set of production rules $u \to v$, where $u, v \in (N \cup T)^*$.

The language of G, denoted by $L(G)$ is the set of terminal strings derivable from the start symbol S. Symbolically, $L(G) = \{w \in T^* \mid S \Rightarrow w\}$. A grammar is accepted only when the produced word contains no auxiliary symbol [6].

Regular grammar, context-free grammar and context-sensitive grammar are among the types of Chomsky grammar. The formal definitions of regular grammar, context-free grammar and context-sensitive grammar are presented respectively in the following.

Definition 7 [8] (Regular Grammar)
A grammar $G = (N, T, S, P)$ is **regular** if the production has the form $u \to v$, where $u \in N$, and $v \in \{\lambda\} \cup T \cup TN$. If v is equal to λ, the production is called a λ-production.

The language produced by such a grammar is called a regular language.

Definition 8 [8] (Context-free Grammar)
A grammar $G = (N, T, S, P)$ is **context-free** if the production has the form $u \to v$, where $u \in N$, and $v \in (N \cup T)^*$. If v is equals λ, the production is called a λ-production.

Context-free grammar will produce a language called a context-free language.

Definition 9 [9] (Context-sensitive Grammar)
A grammar $G = (N, T, S, P)$ is **context-sensitive** if each production has the form $u \to v$, where $u, v \in (N \cup T)^+$, and $\text{length}(u) \leq \text{length}(v)$.

The language produced by context-sensitive grammar is known as the context-sensitive language.

3 PERMUTATION GROUPS AS VALENCES IN SPLICING SYSTEMS

A method of regulated rewriting of rules using valences has been introduced by Paun [10]. An integer (valence) is assigned to the used strings and the integer is computed for a derivation of a new value (new valence).

Extended valence H system over group of integer numbers $(\mathbb{Z},+,0)$ and positive rational numbers $(\mathbb{Q}_+,\cdot,1)$ have been introduced in [4] for the study of generative power of extended valences H systems.

For this research, elements of permutation groups are used as valences in splicing systems to compute the generative power of extended valence H systems over permutation groups.

A definition of extended valence H system over permutation group is defined in the following.

Definition 10 Extended Valence H System Over Permutation Group S_n

Let $(S_n, \cdot, (1))$ be a permutation group S_n with operation, $\cdot$, and the identity, (1). An **extended valence H system over permutation group** S_n is a construct of $\gamma = (V, T, A, R)$, where:

V is an alphabet,

$T \subseteq V$ is the terminal alphabet,

A is a finite subset of $V^* \times S_n$,

$R \subseteq V^*\#V^*\$V^*\#V^*$ is the set of splicing rules.

For $(x,v_1),(y,v_2),(w,v_3) \in V^* \times S_n$ and $r \in R$, where $x, y, w \in V^*$, $v_1, v_2, v_3 \in S_n$, the splicing operation is $[(x,v_1),(y,v_2)] \vdash_r (w,v_3)$, if and only if $(x,y) \vdash_r w$ and $v_3 = v_1 \cdot v_2$. Then $L(\gamma) = \{x \in T^* \mid (x,e) \in \sigma^*(A)\}$.

An element of a permutation group is associated to each axiom, A, and the value of group operation of new strings is computed with each splicing operation. The complete strings produced are considered to be valid if the computation of the associated elements of the group produces the identity element.

Two examples of computations of splicing systems over permutation groups of length two are presented below. First, an example of splicing system over permutation group of length two involving one initial string is shown in the following.

Example 2 Splicing system over permutation group of length two involving one initial string.

Let $(S_2, \cdot, (1))$ be a permutation group of $S_2 = \{(1),(12)\}$ with the multiplicative operation, $\cdot$, and the identity, (1). An extended valence H system over S_2 is a construct of $\gamma = (V, T, A, R)$, where:

$V = \{a,c,d\}$,

$T = \{a,c,d\}$,

A is a finite subset of $V^* \times S_2 = \{[cad,(12)]\}$,

$R \subseteq \{c\#a \$ a\#d\}$ is the set of splicing rules.

By applying $c\#a \$ a\#d$ to initial strings cad, the new strings produced are $w = [\{caad, cd\}, (12)(12)] = [\{caad, cd\}, (1)]$.

The H scheme for this stage is $\sigma^1(A) = \sigma^0(A) \cup \{[caad,(1)], [cd,(1)]\}$. Note that $\sigma^0(A) = A$. Therefore, $\sigma^1(A) = \{[cad,(12)], [caad,(1)], [cd,(1)]\}$.

The next stage of splicing operation is obtained by applying $c\#a \$ a\#d$ to strings $caad$ and cad, and the new strings produced are $w = [\{caaad, cd\}, (1)(12)] = [\{ca^3d, cd\}, (12)]$.

In this stage, the rule $c\#a \$ a\#d$ can also be applied to strings $caad$, and the new strings produced are $w = [\{caaad, cd\}, (1)(1)] = [\{ca^4d, cd\}, (1)]$.

The splicing system for this stage is $\sigma^2(A) = \sigma^1(A) \cup \{[ca^3d,(12)], [cd,(12)], [ca^4d,(1)], [cd,(1)]\}$.

Thus, $\sigma^2(A) = \{[cad,(12)], [caad,(1)], [cd,(1)], [ca^3d,(12)], [cd,(12)], [ca^4d,(1)], [cd,(1)]\}$.

Continuing this splicing process, the resulting language is only accepted if the value of valences is equal to the identity.

Therefore, the language of this extended valence H system is $L(\gamma) = \{ca^{2n}d, n \geq 1\}$. From the Chomsky grammar, the grammars that generate this language are context-sensitive and context-free grammar but not regular. These two grammars are shown in the following.

The generating grammar (context-sensitive grammar) that produced context-sensitive languages:

$G = (\{S, A\}, \{a, c, d\}, S, P)$ with production rules

$$S \rightarrow cAd,$$
$$A \rightarrow aaA \mid aa.$$

The derivations of the language $L(\gamma) = \{ca^{2n}d, n \geq 1\}$ by using context-sensitive grammar are shown in the following.

For n = 1,

$$S \Rightarrow cAd \Rightarrow caad.$$

For n = 2,

$$S \Rightarrow cAd \Rightarrow caaAd \Rightarrow caaaad.$$

For n = 3,

$$S \Rightarrow cAd \Rightarrow caaAd \Rightarrow caaaaAd \Rightarrow caaaaaad.$$

For n a finite number, it is clear that context-sensitive grammars generate $L(\gamma) = \{ca^{2n}d, n \geq 1\}$.

The generating grammar (context-free grammar) that produced context-free languages:

$G = (\{S, A\}, \{a, c, d\}, S, P)$ with production rules

$$S \rightarrow cAd,$$
$$A \rightarrow aaA \mid \lambda.$$

The derivations of the language $L(\gamma) = \{ca^{2n}d, n \geq 1\}$ by using context-sensitive grammar are shown in the following.

For n = 1,

$$S \Rightarrow cAd \Rightarrow caaAd \Rightarrow caa\lambda d \Rightarrow caad.$$

For n = 2,

$$S \Rightarrow cAd \Rightarrow caaAd \Rightarrow caaaaAd \Rightarrow caaaa\lambda d \Rightarrow caaaad.$$

For n = 3,

$$S \Rightarrow cAd \Rightarrow caaAd \Rightarrow caaaaAd \Rightarrow caaaaaaAd \Rightarrow caaaaaa\lambda d \Rightarrow caaaaaad.$$

For n a finite number, it is clear that context-free grammars generate $L(\gamma) = \{ca^{2n}d, n \geq 1\}$.

The two grammars above are not characterized as regular grammars because the production rules of any regular grammar must have either a rightmost variable or a leftmost variable.

Since regular grammar is not characterized in both context-sensitive grammar and context-free grammar, the language $L(\gamma)$ will not produce a regular language. From the Chomsky hierarchy, $L(\gamma) = CS - REG$.

Next, an example of splicing system over permutation group of length two involving two different initial strings is presented.

Example 3 Splicing system over permutation group of length two involving two different initial strings.

Let (S_2, $\cdot$, (1)) be a permutation group of S_2 = {(1),(12)} with the multiplicative operation, $\cdot$, and the identity, (1). An extended valence H system over S_2 is a construct of $\gamma = (V, T, A, R)$, where:

V = $\{a, b, c, d\}$,

$T = \{a, b, c\}$,

A is a finite subset of $V^* \times S_2$ = {[cad,(12)], [dbc,(21]}, $R \subseteq$ {r_1 = $a\#d$ \$ $d\#b$, r_2 = $a\#d$ \$ $c\#a$, r_3 = $b\#c$ \$ $d\#b$} is the set of splicing rules.

By applying $r_1 = a\#d$ \$ $d\#b$ to initial strings cad and dbc, the resulting new strings are w = [{$cabc, dd$}, (12)(21)] = [{$cabc, dd$}, (1)].

Next, by applying $r_2 = a\#d\$c\#a$ to initial strings cad, the resulting new strings are w = [{$caad, cd$}, (12)(12)] = [{$caad, cd$}, (1)].

Then, by applying $r_3 = b\#c\$d\#b$ to initial strings dbc, the resulting new strings are w = [{$dbbc, dc$}, (21)(21)] = [{$dbbc, dc$}, (1)].

The H scheme for this stage is $\sigma^1(A) = \sigma^0(A) \cup$ {[$cabc$,(1)], [dd,(1)], [$caad$,(1)], [cd,(1)], [$dbbc$,(1)], [dc,(1)]}. Note that $\sigma^0(A) = A$. Therefore, $\sigma^1(A)$ = {[cad,(12)], [dbc,(21], [$cabc$,(1)], [dd,(1)], [$caad$,(1)], [cd,(1)], [$dbbc$,(1)], [dc,(1)]}.

Continuing the splicing process to the next stage, and by applying $r_1 = a\#d$ \$ $d\#b$ to strings:

1. $caad$ and $dbbc$: the resulting new strings are w = [{$caabbc, dd$},(1)(1)] = [{ca^2b^2c, dd},(1)].
2. $caad$ and dbc : the resulting new strings are w = [{$caabc, dd$},(1)(21)] = [{ca^2bc, dd},(21)].
3. cad and $dbbc$: the resulting new strings are w = [{$cabbc, dd$},(12)(1)] = [{cab^2c, dd},(12)].

Then, by applying $r_2 = a\#d$ \$ $c\#a$ to strings:

1. $caad$ and cad : the resulting new strings are w = [{$caaad, cd$},(1)(12)] = [{ca^3d, cd},(12)].
2. $caad$: the resulting new strings are w = [{$caaaad, cd$},(12)(12)] = [{ca^4d, cd},(1)].

Next, by applying $r_3 = b\#c\$d\#b$ to strings:

1. $dbbc$ and dbc : the resulting new strings are w = [{$dbbbc, dc$},(1)(21)] = [{db^3c, dc},(21)].
2. $dbbc$: the resulting new strings are w = [{$dbbbbc, dc$},(21)(21)] = [{db^4c, dc},(1)].

Therefore, $\sigma^2(A) = \sigma^1(A) \cup$ {[ca^2b^2c,(1)], [dd,(1)], [ca^2bc,(21)], [dd,(21)], [cab^2c,(12)], [dd,(12)], [ca^3d,(12)], [cd,(12)], [ca^4d,(1)], [cd,(1)], [db^3c,(21)], [dc,(21)], [db^4c,(1)], [dc,(1)]}.

Thus, $\sigma^2(A)$ = {[$cad,(12)$], [$dbc,(21]$, [$cabc,(1)$], [$dd,(1)$], [$ca^2d,(1)$], [$cd,(1)$], [$db^2c,(1)$], [$dc,(1)$], [$ca^2b^2c,(1)$], [$dd,(1)$], [$ca^2bc,(21)$], [$dd,(21)$], [$cab^2c,(12)$], [$dd,(12)$], [$ca^3d,(12)$], [$cd,(12)$], [$ca^4d,(1)$], [$cd,(1)$], [$db^3c,(21)$], [$dc,(21)$], [$db^4c,(1)$], [$dc,(1)$]}.

Continuing this splicing process, the resulting language is only accepted if the value of valences is equal to identity.

The language for this extended valence H system is $L(\gamma) = \{ca^m b^n c,\ m+n = 2k,$ where $m,n,k \geq 1\}$. From the Chomsky grammar, the grammars that generate this language are context-sensitive and context-free grammar but not regular. These two grammars are shown in the following.

The generating grammar (context-sensitive grammar) that produced context-sensitive languages:

$G = (\{S, A, B\}, \{a, b, c\}, S, P)$ with production rules

$$S \rightarrow cABc,$$
$$A \rightarrow aaA \mid abB,$$
$$B \rightarrow bbB.$$

The derivations of the language $L(\gamma) = \{ca^m b^n c,\ m+\ \ n = 2k$ where $m,\ n,\ k \geq 1\}$ by using context-sensitive grammar are shown in the following.

Case 1: For k = 1, when m, n = 1:

$$S \Rightarrow cAc \Rightarrow cabc.$$

Case 2: For k = 2, when m = 1 and n = 3:

$$S \Rightarrow cAc \Rightarrow cabBc \Rightarrow cabbbc.$$

Case 3: For k = 3, when m = 2 and n = 4:

$$S \Rightarrow cAc \Rightarrow caaBc \Rightarrow caabbBc \Rightarrow caabbbbc.$$

Case 4: For k = 4, when m = 6 and n = 2:

$$S \Rightarrow cAc \Rightarrow caaAc \Rightarrow caaaaAc \Rightarrow caaaaaaBc \Rightarrow caaaaaabbc.$$

For m, n, and k finite numbers, it is clear that context-sensitive grammars generate $L(\gamma) = \{ca^m b^n c,\ m+n = 2k$ where $m, n, k \geq 1\}$.

The generating grammar (context-free grammar) that produced context-free languages:

$G = (\{S, A, B\}, \{a, b, c\}, S, P)$ with production rules

$$S \rightarrow cABc,$$
$$A \rightarrow aaA \mid abB,$$
$$B \rightarrow bbB \mid \lambda.$$

The derivations of the language $L(\gamma) = \{ca^m b^n c, m+n = 2k$ where $m, n, k \geq 1\}$ by using context-sensitive grammar are shown in the following.

Case 1: For k = 1, when m, n = 1:

$$S \Rightarrow cAc \Rightarrow cabBc \Rightarrow cab\lambda c \Rightarrow cabc.$$

Case 2: For k = 2, when m = 1 and n = 3:

$$S \Rightarrow cAc \Rightarrow cabBc \Rightarrow cabbbBc \Rightarrow cabbb\lambda c \Rightarrow cabbbc.$$

Case 3: For k = 3, when m = 2 and n = 4:

$$S \Rightarrow cAc \Rightarrow caaBc \Rightarrow caabbBc \Rightarrow caabb\lambda c \Rightarrow caabbbbc.$$

Case 4: For k = 4, when m = 6 and n = 2:

$$S \Rightarrow cAc \Rightarrow caaAc \Rightarrow caaaaAc \Rightarrow caaaaaaBc \Rightarrow caaaaaabbBc \Rightarrow caaaaaabb\lambda c \Rightarrow caaaaaabbc.$$

For m, n, and k finite numbers, it is clear that context-free grammars generate $L(\gamma) = \{ca^m b^n c,\ m+n = 2k$ where $m, n, k \geq 1\}$.

The two grammars above are not characterized as regular grammars because the production rules of any regular grammar must have either a rightmost variable or a leftmost variable.

Since regular grammar is not characterized in both context-sensitive grammar and context-free grammar, the language $L(\gamma)$ will not produce a regular language. From the Chomsky hierarchy, $L(\gamma) = CS - REG$.

From the two examples, some splicing systems over permutation groups of length two involving one initial string and two different initial strings have increased the generative power of splicing systems since they do not only generate the regular languages. Therefore, the results of this research show that splicing systems over some permutation groups have increased the generative power of splicing systems.

4 CONCLUSION

This paper initiates the study of permutation groups in splicing systems. Since splicing systems only generate regular languages, the use of valences increases the generative power of splicing systems up to context-sensitive languages. The results of this paper show that splicing systems over some permutation groups have increased the generative power of extended splicing systems.

ACKNOWLEDGEMENT

The authors would like to acknowledge Research Management Center (RMC), UTM for the partial financial funding under Research University Fund Vote No. 02J65. The first and second authors would like to thank Ministry

of Higher Education (MOHE) for MyMaster scholarship programme under the Tenth Malaysia Plan (10MP).

REFERENCES

[1] 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.

[2] D. Pixton, Regularity of Splicing Languages, Discrete Applied Mathematics, 69(1996), 101–124.

[3] R.W. Gatterdam, Splicing Systems and Regularity, International Journal of Computer Math., 31(1989), 63–67.

[4] V. Manca, and G. Paun, Arithmetically Controlled H Systems, Computer Science Journal of Moldova, 6(1998), 103–118.

[5] G. Paun, G. Rozernberg, and A. Salomaa, DNA Computing: New Computing Paradigms, Springer-Verlag, Heidelberg, 1998.

[6] R. Freund, L. Kari, and G. Paun, The Existence of Universal Computers, Theory of Computing Systems, 32(1999), 69 –112.

[7] J. Fraleigh, A First Course in Abstract Algebra, Addison-Wesley Publishing Company Inc., Philippines, 1997.

[8] N. Vugt, Models of Molecular Computing, PhD Thesis, Leiden University, 2002.

[9] T. A. Sudkamp, Languages and Machines: An Introduction to the Theory of Computer Science, Addison-Wesley Longman Inc., California, 1997.

[10] H. Fernau, and R. Stiebe, Regulation by Valences, in Rovan, B. (Ed.), Mathematical Foundations of Computer Science 1997 22nd International 47 Symposium (MFCS '97) Proceedings, Springer-Verlag, Slovakia, 129 (1997), 239–248.