Volume 13 Number 4

ACTA CYBERNETICA

Szeged, 1998

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Contributions are accepted for review with the understanding that the same work has not been published elsewhere.

Manuscripts must be in English and should be sent in triplicate to any of the Editors. On the first page, the *title* of the paper, the *name(s)* and *affiliation(s)*, together with the *mailing* and *electronic address(es)* of the author(s) must appear. An *abstract* summarizing the results of the paper is also required. References should be listed in alphabetical order at the end of the paper in the form which can be seen in any article already published in the journal. Manuscripts are expected to be made with a great care. If typewritten, they should be typed double-spaced on one side of each sheet. Authors are encouraged to use any available dialect of TEX.

After acceptance, the authors will be asked to send the manuscript's source TEX file, if any, on a diskette to the Managing Editor. Having the TEX file of the paper can speed up the process of the publication considerably. Authors of accepted contributions may be asked to send the original drawings or computer outputs of figures appearing in the paper. In order to make a photographic reproduction possible, drawings of such figures should be on separate sheets, in India ink, and carefully lettered.

There are no page charges. Fifty reprints are supplied for each article published.

Publication information. Acta Cybernetica (ISSN 0324–721X) is published by the Department of Informatics of the József Attila University, Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. For 1998 Numbers 3–4 of Volume 13 are scheduled. Subscription prices are available upon request from the publisher. Issues are sent normally by surface mail except to overseas countries where air delivery is ensured. Claims for missing issues are accepted within six months of our publication date. Please address all requests for subscription information to: Department of Informatics, József Attila University, H-6701 Szeged, P.O.Box 652, Hungary. Tel.: (36)-(62)-420-184, Fax:(36)-(62)-420-292.

URL access. All these information and the contents of the last some issues are available in the Acta Cybernetica home page at http://www.inf.u-szeged.hu/local/acta.

H. Bunke
Universität Bern
Institut für Informatik und
angewandte Mathematik
Längass strasse 51., CH-3012 Bern
Switzerland

B. Courcelle
Universitè Bordeaux-1
LaBRI, 351 Cours de la Libèration
33405 TALENCE Cedex
France

J. Demetrovics
MTA SZTAKI
Budapest, P.O.Box 63
H-1502 Hungary

B. Dömölki
IQSOFT
Budapest, Teleki Blanka u. 15-17.
H-1142 Hungary

J. Engelfriet
Leiden University
Computer Science Department
P.O. Box 9512, 2300 RA Leiden
The Netherlands

Z. Ésik
A. József University
Department of Foundations of
Computer Science
Szeged, Aradi vértanúk tere 1.
H-6720 Hungary

L. Lovász
Eötvös Loránd University
Budapest Múzeum krt. 6-8.
H-1088 Hungary

G. Păun
Institute of Mathematics
Romanian Academy
P.O.Box 1-764, RO-70700
Bucuresti, Romania

A. Prékopa
Eötvös Loránd University
Budapest, Múzeum krt. 6-8.
H-1088 Hungary

A. Salomaa
University of Turku
Department of Mathematics
SF-20500 Turku 50, Finland

L. Varga
Eötvös Loránd University
Budapest, Múzeum krt. 6-8.
H-1088 Hungary

H. Vogler
Dresden University of Technology
Department of Computer Science
Foundations of Programming
D-01062 Dresden, Germany

G. Wöginger
Technische Universität Graz
Institut für Mathematik (501B)
Steyrergasse 30
A-8010 Graz, Österreich

Acta Cybernetica **13** (1998) 339-358.

Grammars Working on Layered Strings

Paolo Bottoni, * Giancarlo Mauri, † Piero Mussio, ‡
Gheorghe Păun §

Abstract

We consider first an operation with strings and languages suggested by superposed windows on the computer screen (as well as by cryptographic systems of Richelieu type): we assume that the strings contain usual symbols as well as a transparent symbol. Superposing two strings (justified to left), we produce a new string consisting of the symbols observable from above. This operation is investigated as an abstract operation on strings, then it is used in building a variant of grammar systems with the component grammars working on the layers of an array of strings. Each grammar can rewrite only symbols in its layer which are observable from above. The language generated in this way consists of strings of the observable symbols, produced at the end of a derivation. The power of several variants of these generative mechanisms is investigated for the case of two layered strings. When a matrix-like control on the work of the component grammars is considered, then a characterization of recursively enumerable languages is obtained.

1 Introduction

Recent work in the study of algebraic features of pictorial languages has shown how a natural operation between pictures is that of superposition. This operation is informally understood as the placing of one image (which can contain some *transparent* symbols) above another, so that only pixels in the second image which appear immediately below transparent pixels in the first image are *observable* and can contribute to the resulting image [2]. Actually, superposition of structures on the screen is continuously used in visual interactive systems. Consider for example windowing systems in which windows are allowed to overlap, as in the MacOs^{TM} and Windows^{TM} operating systems. In these cases what is actually observable by a

*Department of Computer Science, University of Rome "La Sapienza", Via Salaria 113, 00198 Roma, Italy, e-mail:bottoni@dsi.uniroma1.

†Department of Computer Science, University of Milano, Via Comelico 39, 20135 Milano, Italy, e-mail:mauri@dsi.unimi.it.

‡Department of Electronics for Automation, University of Brescia, Via Branze 38, 25123 Brescia, Italy, e-mail:mussio@bsing.ing.unibs.it.

§Institute of Mathematics of the Romanian Academy, PO Box 1 - 764, 70700 Bucureşti, Romania, e-mail:gpaun@imar.ro.

user results from the spatial relations among the windows currently on the screen. If one considers that each window contains a sentence in a language, the screen defines a sentence which results from the superposition of several such partial sentences. The study of the characteristics of such languages, and hence of the properties of the superposition operation becomes interesting if one wants to avoid situations which may generate disorientation in the user [3].

The idea of employing transparency as a language-defining tool predates the appearance of computers of a long time. It can be traced back to the Richelieu code, an elementary form of cryptography in which a message is embedded in a random text, and can be recovered by superposing the whole text by an opaque sheet with holes in it. The letters reading through the holes form the original message [1]. On the other hand, non-transparent pixels are those on which the user can interact, hence are those from which the transformations of the current sentence may originate. This suggests a notion of control in which the activation of symbols in a rewriting process is possible only if such symbols are *visible*, i.e., observable and non-transparent. Such a form of control appears to be very frequent in natural and artificial systems, in all cases in which layers may be defined and the development of a phenomenon depends on the characteristics both of the layer at which it occurs and of the above layers (e.g., rain permeability of a terrain, the growth of films in VLSI chips, competition for light in chlorophylliac plants, radiology methods based on differences in tissue absorbing properties, etc.). The notion of layer is also useful in defining systems in which different agents may cooperate to define the evolution of a substrate. Different systems may have different roles and be allowed to operate only on parts left undefined by prominent agents. A similar model was also at the basis of the proposal of Parallel Communicating Grammar Systems, where query symbols are used by a master agent to mark the places where other, specified, agents may contribute strings of unknown length [9]. In the notion of layered grammar proposed in this paper, instead, agents can operate autonomously and independently, but their contribution to the final result is limited to fill transparent "holes" left by an agent in an upper position. Which agent will contribute in which zones of the substrate to the final result cannot be established a priori, depending on the ability of an agent to synchronize its activity with that of agents at an upper layer.

Hence, two problems appear worth studying related to the notion of layer. First, what are the properties of the superposition operation and which properties and families of languages it preserves. Second, which is the expressive power of layers as a modelling tool. We start this study by considering closure properties of the operation and by exploring the generative power of context-free grammars with only two layers. It results that by such simple tools, more precisely by using layered right-linear grammars with a specific form of synchronization, we can provide a characterization of recursively enumerable languages. Further study can be performed on the characteristics of systems with more than two layers and on an extension of the notion of transparency (for instance by considering symbols with different levels of opacity).

2 Formal Language Theory Prerequisites

In this section we recall only a few notions, notations, and results needed below. Further details can be found in [11] and in references therein. For an alphabet V, we denote by V^* the free monoid generated by V; λ is the empty string, $|x|$ is the length of $x \in V^*$. The language of the non-empty strings over V, that is $V^* - \{\lambda\}$, is denoted by V^+. A morphism $h : V^* \longrightarrow U^*$ such that $h(a) \in U \cup \{\lambda\}$ for all $a \in V$ is called a *weak coding*; it is called a *coding* when $h(a) \in U$ for all $a \in V$ and a *projection* when $h(a) \in \{a, \lambda\}$ for all $a \in V$. A string $x \in V^*$ can be seen as a mapping $x : \{1, 2, \ldots, \infty\} \longrightarrow V \cup \{\#\}$, where $\#$ is the *blank* symbol, with the following properties: there is $i \geq 1$ such that $x(i) = \#$; moreover, if $x(j) = \#$, then $x(j+1) = \#, j \geq 1$ (this means that $x(j) \in V$ for $1 \leq j \leq |x|$ and $x(j) = \#$ otherwise). Sometimes, we shall use below such an interpretation of a string. A Chomsky grammar is a construct $G = (N, T, S, P)$, where N, T are disjoint alphabets, $S \in N$, and P is a finite subset of $(N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$. The elements of N are called nonterminal symbols, those of T are called terminal symbols, S is the axiom, and the elements $(u, v) \in P$, written in the form $u \to v$, are called rewriting rules (for short, rules). The language generated by G is denoted $L(G)$. When all rules in P are of the form $A \to x, A \in N, x \in (N \cup T)^*$, then the grammar is said to be *context-free*; it is *linear* if x above contains at most one nonterminal symbol. A context-free grammar whose rules are of the forms $A \to xB, A \to x$, for $A, B \in N, x \in T^*$, is said to be *right-linear*; when x above is a single symbol in T, then the grammar is said to be *regular*. We denote by $FIN, REG, LIN, CF, CS, RE$ the families of finite, regular, linear, context-free, context-sensitive, and recursively enumerable languages, respectively.

For $x, y \in V^*$, we define the *shuffle* of x, y by

$$\begin{aligned} x \amalg y \;=\; & \{x_1 y_1 x_2 y_2 \ldots x_n y_n \mid n \geq 1, x = x_1 x_2 \ldots x_n, \\ & y = y_1 y_2 \ldots y_n, x_i, y_i \in V^*, 1 \leq i \leq n\}. \end{aligned}$$

The concatenation has priority over shuffle: $L_1 L_2 \amalg L_3$ should be read as $(L_1 L_2) \amalg L_3$.

Convention. Two language generating mechanisms are considered equivalent if they generate languages which differ at most by the empty string.

3 The Operation of Superposition

In what follows, t is always a special symbol, which is considered *transparent*. Let V be an alphabet. Two strings $x, y \in (V \cup \{t\})^*$ can be *superposed*, producing a third string z which is constructed as follows:

1. the shortest of x, y is completed to the right with occurrences of t such that two strings of the same length are obtained; let us denote by x', y' the strings obtained in this way (at least one of them is not modified);

2. then, for $i \geq 1$, we set

$$z(i) = \begin{cases} x'(i), & \text{if } x'(i) \neq t, \\ y'(i), & \text{otherwise.} \end{cases}$$

(Clearly, $|z| = \max\{|x|, |y|\}$.) We denote the string z by $x \diamond y$ and we say that it is obtained by the *superposition* of x and y. We can imagine that $x \diamond y$ is obtained by writing the strings x, y one over the other, with x above, aligned to left, and looking from above to the "layered string" obtained in this way. Through transparent symbols we observe the corresponding symbols of y (maybe also the transparent symbol), otherwise we observe the symbols of x.

For **example**, for

$$x = abttabt, \quad y = tabttbbabt,$$

we obtain

$$x \diamond y = abb\underline{t}ab\underline{babt},$$

where the underlined symbols are taken from y.

Obviously, the operation $\diamond$ is associative, but not commutative.

For $L_1, L_2 \subseteq (V \cup \{t\})^*$ we put

$$L_1 \diamond L_2 = \{x \diamond y \mid x \in L_1, y \in L_2\}.$$

Note that *every* string from L_1 is superposed with *every* string in L_2, irrespective of their length, because of the completion with occurrences of t of the shortest string in each pair.

We now investigate the operation of superposition as an abstract operation on languages, relating it to other operations on languages. In this way, the closure properties of families in the Chomsky hierarchy under this new operation will be settled.

Since our goal is the study of the closure properties of language families in the Chomsky hierarchy, all families we consider below are supposed to contain at least all regular languages (but this is not stated again and again).

Lemma 1. *If F is a family of languages which is closed under superposition, shuffle with regular languages, weak codings, and intersection with regular languages, then it is closed under intersection.*

Proof. Consider two languages $L_1, L_2 \subseteq V^*$. Denote $V' = \{a' \mid a \in V\}$, where a' is a new symbol associated with $a \in V$, and define the weak coding $h_1 : (V \cup V')^* \longrightarrow V^*$ by $h_1(a') = \lambda, a' \in V'$, and $h_1(a) = a, a \in V$, and the coding $h_2 : (V \cup \{t\})^* \longrightarrow (V' \cup \{t\})^*$ by $h_2(a) = a', a \in V$, and $h_2(t) = t$. Consider also the regular languages

$$R_1 = \{at \mid a \in V\}^*,$$
$$R_2 = \{ta \mid a \in V\}^*,$$
$$R_3 = \{aa' \mid a \in V\}^*.$$

Then the following equality holds

$$L_1 \cap L_2 = h_1((L_1' \diamond L_2') \cap R_3),$$

where

$$L_1' = (L_1 \sqcup t^*) \cap R_1,$$
$$L_2' = h_2((L_2 \sqcup t^*) \cap R_2).$$

(The intersection with R_1, R_2 forces the shuffling with t^* to pair symbols of strings in L_1, L_2 with symbols t, then the intersection with R_3 selects only those strings coresponding to equal strings from L_1, L_2. Finally, the weak coding h_1 discards symbols in V'.) In view of the closure properties of the family F, it follows that F is closed under intersection. □

Corollary 1. *The families LIN, CF are not closed under superposition.*

Lemma 2. *If F is a family of languages which is closed under intersection with regular languages, shuffle, codings, and inverse morphisms, then F is closed under superposition.*

Proof. Consider two languages $L_1, L_2 \subseteq (V \cup \{t\})^*$, denote as above $V' = \{a' \mid a \in V\}$, consider the new symbols c, c', t' and the new alphabet

$$\begin{aligned} U &= \{[ab'], [at'], [ac'], [ca'], [ta'] \mid a, b \in V\} \\ &\cup \{[tt'], [ct'], [tc']\}, \end{aligned}$$

and define the following morphisms:

$h_1 : U^* \longrightarrow V^*$,
by $h_1([ab']) = h_1([at']) = h_1([ta']) = h_1([ca']) = h_1([ac']) = a$, for $a, b \in V$,
and $h_1([tt']) = h_1([ct']) = h_1([tc']) = t$,
$h_2 : U^* \longrightarrow (V \cup V' \cup \{t, t', c, c'\})^*$,
by $h_2([\alpha\beta]) = \alpha\beta$, for $[\alpha\beta] \in U$,
$h_3 : (V \cup \{t\})^* \longrightarrow (V' \cup \{t'\})^*$,
by $h_3(a) = a'$, $a \in V$, and $h_3(t) = t'$.

With the regular language

$$R = [(V \cup \{t\})(V' \cup \{t'\})]^*[((V \cup \{t\})\{c'\})^* \cup (\{c\}(V' \cup \{t'\}))^*]$$

we obtain

$$L_1 \diamond L_2 = h_1(h_2^{-1}((L_1\{c\}^* \sqcup h_3(L_2)\{c'\}^*) \cap R)).$$

The intersection with R selects, from the strings produced by shuffling, those strings which are obtained by interleaving the symbols of strings in L_1, L_2, maybe prolonged with occurrences of c, c', but not containing superfluous occurrences of c, c';

then, h_2^{-1} replaces blocks $\alpha\beta$ by symbols $[\alpha\beta]$ which are "interpreted" by h_1 in the same way as when constructing the superposition of the two strings in L_1, L_2. The use of symbols c, c' prevents the addition of superfluous symbols t in the right end of strings. From the closure properties of F, we get $L_1 \diamond L_2 \in F$.

Note that, by our convention above, $\{c\}^* \in REG \subseteq F$ and $L\{c\}^* = (L \sqcup\!\sqcup \{c\}^*) \cap V^*\{c\}^*$, that is, F is closed under concatenation with $\{c\}^*$. (In fact, by the closure properties of F, we can get the closure of F under concatenation with any regular language, but we do not need here this general property.) □

Corollary 2. *The families REG, CS, RE are closed under the superposition.*

Corollary 3. *If F is a family of languages which is closed under intersection with regular languages, shuffle with regular languages, codings, and inverse morphisms, then F is closed under superposition with regular languages, in the following sense: if $L_1 \in F, L_2 \in REG$, then $L_1 \diamond L_2 \in F$ and $L_2 \diamond L_1 \in F$.*

Proof. If one of the languages L_1, L_2 is regular, then in the proof above we use a shuffle with regular languages. □

The families LIN, CF are closed under shuffle with regular languages, hence they are closed under superposition with regular languages.

4 Grammars Working on Layered Strings

For two strings $x, y \in (V \cup \{t\})^*$ we denote by $[x, y]$ the two-level sequence obtained by placing x over y, justified to left and completing the shortest string with occurrences of t; $[x, y]$ is called a *layered string*. Given a layered string $[x, y]$, any symbol $x(i) \in V$ is said to be *observable*. A symbol $y(i) \in V \cup \{t\}$ is *observable* if and only if $x(i) = t$. If $y(i) \in V$, then $y(i)$ is also *visible*. Therefore, the observable symbols in $[x, y]$ correspond to the symbols appearing in the string $x \diamond y$ defined as in the previous section. In the sequel, for simplicity, we will only use the term observable. We can now define the main notion investigated in this paper.

A *layered grammar* is a construct

$$\gamma = (N, T, t, (S_1, P_1), (S_2, P_2)),$$

where N, T are disjoint alphabets, t is a special symbol not in $N \cup T$, $S_1, S_2 \in N$, and P_1, P_2 are finite sets of context-free rules over $N \cup T \cup \{t\}$ (t is considered a terminal symbol); N is the *nonterminal* alphabet, T is the *terminal* alphabet, t is the transparent symbol, (S_i, P_i) are the *components* of the grammar; S_i is the *axiom* and P_i is the set of rules of component $i, i = 1, 2$. We also say that (S_1, P_1) is the upper component and (S_2, P_2) is the lower component of γ.

For $x_1, x_2, y_1, y_2 \in (N \cup T \cup \{t\})^*$ we write $[x_1, x_2] \Longrightarrow_s [y_1, y_2]$ if and only if both the following conditions hold:

(1) $x_1 = x_1' A x_1'', y_1 = x_1' u_1 x_1'', A \to u_1 \in P_1$, or

$$
\begin{array}{ll}
 & x_1 = y_1 \in (T \cup \{t\})^*, \\
(2) & x_2 = x_2' A x_2'', y_2 = x_2' u_2 x_2'', A \to u_2 \in P_2, A \text{ is observable, or} \\
 & x_2 = y_2 \text{ and no nonterminal symbol is observable in } x_2.
\end{array}
$$

The relation $\Longrightarrow_s$ is called a *synchronized derivation* in γ: each component has to use a rule, except the case when the corresponding string is terminal, or, in the case of the lower component, no nonterminal is observable. Therefore, only the observable symbols of the lower level are active and can be rewriten. A variant of the relation $\Longrightarrow_s$ is $\Longrightarrow_{ns}$, the *non-synchronized* derivation step: for $x_1, x_2, y_1, y_2 \in (N \cup T \cup \{t\})^*$ we write $[x_1, x_2] \Longrightarrow_{ns} [y_1, y_2]$ if and only if one of the following cases holds:

$$
\begin{array}{ll}
(1) & x_1 = x_1' A x_1'', y_1 = x_1' u_1 x_1'', A \to u_1 \in P_1, \text{ and} \\
 & x_2 = y_2, \\
(2) & x_1 = y_1 \text{ and} \\
 & x_2 = x_2' A x_2'', y_2 = x_2' u_2 x_2'', A \to u_2 \in P_2, A \text{ is observable.}
\end{array}
$$

Only one of the two components works, rewriting any symbol in the upper level and an observable symbol in the lower level. By rewriting first in the lower level and then in the upper level we can simulate in this way a synchronized derivation. Thus, $\Longrightarrow_s$ is indeed a restricted version of $\Longrightarrow_{ns}$. For any $\Longrightarrow_\alpha, \alpha \in \{s, ns\}$ we denote by $\Longrightarrow_\alpha^*$ its reflexive and transitive closure. For each relation $\Longrightarrow_\alpha$ we can consider two languages associated to γ, by considering two stop conditions for a derivation: when no nonterminal is allowed in the last layered string, we get

$$
L_{t,\alpha}(\gamma) = \{z_1 \diamond z_2 \in (T \cup \{t\})^* \mid [S_1, S_2] \Longrightarrow_\alpha^* [z_1, z_2],\ z_1, z_2 \in (T \cup \{t\})^*\}.
$$

When we allow finishing the derivation with non-observable nonterminal symbols in the lower level, then we get

$$
\begin{array}{rcl}
L_{nt,\alpha}(\gamma) & = & \{z_1 \diamond z_2 \in (T \cup \{t\})^* \mid [S_1, S_2] \Longrightarrow_\alpha^* [z_1, z_2],\ z_1 \in (T \cup \{t\})^*, \\
 & & z_2 \in (N \cup T \cup \{t\})^*, \text{ but no nonterminal in } z_2 \text{ is observable}\}.
\end{array}
$$

In both cases, $\alpha \in \{s, ns\}$. In this way, we associate with γ four languages, $L_{t,s}(\gamma)$, $L_{t,ns}(\gamma)$, $L_{nt,s}(\gamma)$, $L_{nt,ns}(\gamma)$. We denote by $TSL(X), TNSL(X), NTSL(X), NTNSL(X)$ the families of languages of these types generated by layered grammars with rules of type X; for X we consider here REG, RL, LIN, CF (regular, right-linear, linear, context-free, respectively); we do not distinguish between grammars allowed to contain λ-rules and λ-free grammars (that is, we allow erasing rules). In the proofs in the following section we shall present several specific layered grammars, hence we do not give here examples.

5 Preliminary Results

Directly from the definitions, we obtain

Lemma 3. *$YL(REG) \subseteq YL(RL) \subseteq YL(LIN) \subseteq YL(CF)$, for all $Y \in \{TS, TNS, NTS, NTNS\}$.*

By adding to each set P_1, P_2 rules $A \to A$ for each $A \in N$, we can simulate a non-synchronized derivation by a synchronized one, hence we get

Lemma 4. *$TNSL(X) \subseteq TSL(X), NTNSL(X) \subseteq NTSL(X)$, for each $X \in \{RL, LIN, CF\}$.*

The use of chain rules is important here. We shall see below that layered grammars with regular rules generate only regular languages, both in the synchronized and the non-synchronized modes, hence the result above holds in this indirect way also for the regular case.

Lemma 5. *$X \subseteq YL(X)$, for all $X \in \{REG, RL, LIN, CF\}$, $Y \in \{TS, TNS, NTS, NTNS\}$.*

Proof. For a usual grammar $G = (N, T, S, P)$ we construct the layered grammar

$$\gamma = (N \cup \{S_1\}, T, t, (S_1, \{S_1 \to t\}), (S, P)).$$

Because the upper component generates only one transparent symbol and then stops, we obviously obtain $L_{t,\alpha}(\gamma) = L_{nt,\alpha}(\gamma) = L(G), \alpha \in \{s, ns\}$. □

There is a close relation between the work of layered grammars and the superposition operation (between the superposition of context-free languages and languages generated by a layered grammar). The next result illustrates this.

Theorem 1. *For all context-free languages $L_1, L_2 \subseteq (T \cup \{t\})^*$ there are a weak coding h, a regular language R, and a layered context-free grammar γ such that*

$$L_1 \diamond L_2 = h(L_{t,ns}(\gamma) \cap R).$$

Proof. Let $G_i = (N_i, T \cup \{t\}, S_i, P_i)$ be two context-free grammars with $N_1 \cap N_2 = \emptyset$ such that $L_i = L(G_i), i = 1, 2$. Let S_1', S_2' and A be new nonterminals and let c_1, c_2 be new terminals. We construct the layered grammar

$$\gamma = (N', T', t, (S_1', P_1'), (S_2', P_2')),$$

with

$$\begin{aligned} N' &= N_1 \cup N_2 \cup \{S_1', S_2', A\}, \\ T' &= T \cup \{c_1, c_2, t'\}, \\ P_1' &= \{S_1' \to tS_1',\ S_1' \to c_1 S_1\} \cup P_1, \\ P_2' &= \{S_2' \to AS_2,\ A \to t'A,\ A \to c_2 t\} \cup P_2. \end{aligned}$$

Consider also the weak coding $h : T'^* \longrightarrow (T \cup \{t\})^*$ defined by $h(a) = a, a \in T$, $h(t') = h(c_1) = h(c_2) = \lambda$, as well as the regular language

$$R = \{t'\}^* \{c_2 c_1\} (T \cup \{t\})^*.$$

We obtain the equality $L_1 \diamond L_2 = h(L_{t,ns}(\gamma) \cap R)$. Indeed, the derivations in γ leading to strings in R are equivalent (modulo the order of some steps) to a derivation of the following form:

$$\begin{aligned}
[S_1', S_2'] &\Longrightarrow_{ns}^{*} [t^n S_1', S_2'] \Longrightarrow_{ns} [t^n c_1 S_1, S_2'], \text{ for } n \geq 1, \\
&\Longrightarrow_{ns}^{*} [t^n c_1 w_1, S_2'], \text{ for } w_1 \in L_1, \\
&\Longrightarrow_{ns} [t^n c_1 w_1, AS_2] \Longrightarrow_{ns}^{*} [t^n c_1 w_1, Aw_2], \text{ for } w_2 \in L_2, |w_2| \leq n-1, \\
&\Longrightarrow_{ns}^{*} [t^n c_1 w_1, t'^{n-1} c_2 t w_2].
\end{aligned}$$

Clearly, $(t^n c_1 w_1) \diamond (t'^{n-1} c_2 w_2) = t'^{n-1} c_2 c_1 (w_1 \diamond w_2)$. With the weak coding h, we obtain the string $w_1 \diamond w_2$. □

Corollary 4. $TNSL(CF) - CF \neq \emptyset$, $TSL(CF) - CF \neq \emptyset$.

Proof. The family CF is not closed under the operation $\diamond$, but it is closed under intersection with regular languages and arbitrary morphisms. Moreover, $TNSL(CF) \subseteq TSL(CF)$ (Lemma 4). □

We shall strenghten this result in the following section.

6 The Power of Layered Grammars

First, we show that all families of languages generated by layered grammars with linear rules can generate non-context-free languages. (Note that this does not follow from the proof of Theorem 1, because we use the non-linear rule $S_2' \to AS_2$ in P_2'.)

Theorem 2. $YL(LIN) - CF \neq \emptyset$, $Y \in \{TS, TNS, NTS, NTNS\}$.

Proof. Let us consider the following layered grammar

$$\begin{aligned}
&\gamma = (\{S_1, S_1', S_1'', S_2, S_2'\}, \{a, b, c, d, e\}, t, (S_1, P_1), (S_2, P_2)), \\
&P_1 = \{S_1 \to tS_1',\ S_1' \to S_1''d,\ S_1'' \to aS_1''t,\ S_1'' \to et\}, \\
&P_2 = \{S_2 \to ttS_2,\ S_2 \to tS_2,\ S_2 \to dS_2',\ S_2' \to bS_2'c,\ S_2' \to te\}.
\end{aligned}$$

A non-synchronized derivation in γ is equivalent (modulo the order of some steps) to a derivation of the following form. After using the rule $S_1 \to tS_1'$ in the upper layer, the rule $S_2 \to ttS_2$ must be used in the lower one in order to "get free" this component:

$$[S_1, S_2] \Longrightarrow_{ns} [tS_1', S_2] \Longrightarrow_{ns} [tS_1', ttS_2].$$

In the second component we can derive freely:

$$\begin{aligned}
&[tS_1', ttS_2] \Longrightarrow_{ns}^{*} [tS_1', t^m S_2], \text{ for } m \geq 2, \\
&\Longrightarrow_{ns} [tS_1', t^m dS_2'] \Longrightarrow_{ns}^{*} [tS_1', t^m db^n S_2' c^n], \text{ for } n \geq 0, \\
&\Longrightarrow_{ns} [tS_1', t^m db^n tec^n].
\end{aligned}$$

At any time, we can also start deriving in the upper component, where a string $ta^r ett^r d, r \geq 0$, can be produced. If we consider also the regular language

$$R = ta^+ edb^+ dec^+,$$

and we look only for strings in $L_{t,ns}(\gamma) \cap R$, then we have to stop by producing a layered string

$$[ta^r ett^r d, t^m db^n tec^n],$$

such that $r + 2 = m, r = n$. Therefore, the obtained string is

$$(ta^r ett^r d) \diamond (t^m db^n tec^n) = ta^n edb^n dec^n.$$

Consequently,

$$L_{t,ns}(\gamma) \cap R = \{ta^n edb^n dec^n \mid n \geq 0\},$$

which is not a context-free language. Because the intersection with R asks for having the terminal rule $S_2' \to te$ used, we have $L_{t,ns}(\gamma) \cap R = L_{nt,ns}(\gamma) \cap R$. Therefore, $TNSL(LIN) - CF \neq \emptyset, NTNSL(LIN) - CF \neq \emptyset$. With Lemma 4, also $TSL(LIN) - CF \neq \emptyset, NTSL(LIN) - CF \neq \emptyset$. □

We consider now the case of right-linear layered grammars. When they work in the synchronized mode, they can generate non-regular languages. (However, we do not know whether or not also non-context-free languages can be generated in this way.)

Theorem 3. $TSL(RL) - REG \neq \emptyset$, $NTSL(RL) - REG \neq \emptyset$.

Proof. Consider the layered grammar

$$\gamma = (\{S_1, S_1', S_2\}, \{a, b, c, d\}, t, (S_1, P_1), (S_2, P_2)),$$
$$P_1 = \{S_1 \to tS_1,\ S_1 \to b^2 tb^2 S_1',\ S_1' \to b^2 tb^2 S_1',\ S_1' \to b^2 tc\},$$
$$P_2 = \{S_2 \to a^3 S_2, S_2 \to a^3 d\},$$

as well as the regular language

$$R = a^+ (bbabb)^+ bbdc.$$

In order that a terminal synchronized derivation in γ will produce a string in R, it has to proceed as follows. After the first step, S_2 is observable; it runs faster than S_1 as long as the upper component uses the rule $S_1 \to tS_1$; after starting to use another rule, the upper component runs faster and eventually it catches up the lower one. This must indeed happen when looking for strings in R, because we must obtain a string containing both the symbol c (introduced by the upper component) and the symbol d (introduced by the lower component), in neighboring positions. Thus, we have to follow derivations of the form

$$\begin{aligned}[S_1, S_2] &\Longrightarrow_s [tS_1, S_2] \Longrightarrow_s^* [tt^n S_1, a^{3n} S_2], \text{ for } n \geq 0, \\ &\Longrightarrow_s [tt^n bbtbbS_1', a^{3n} a^3 S_2] \\ &\Longrightarrow_s^* [tt^n (bbtbb)^m S_1', a^{3n} a^{3m} S_2], \text{ for } m \geq 0.\end{aligned}$$

The second component can stop in any moment, but the upper one must derive until catching up (in order to produce the substring dc). Therefore, we have to produce a layered string of the form

$$\Longrightarrow_s^* [tt^n(bbtbb)^{m+p}bbtc, a^{3(n+m)}d],\ p \geq 0,$$

such that

$$n + 1 + 5(m + p) + 3 = 3(n + m) + 1 \tag{1}$$

(in order to have d adjacent to c). This implies that $2m + 5p + 3 = 2n$; because $p \geq 0$, we obtain

$$2m + 3 \leq 2n. \tag{2}$$

Consequently, we get

$$L_{t,s}(\gamma) \cap R = \{a^{n+1}(bbabb)^m bbdc \mid n \geq 0, 2n \geq 2m + 3\}.$$

The intersection $L_{t,s}(\gamma) \cap R$ is infinite. More precisely, this intersection contains strings $a^{n+1}(bbabb)^m bbdc$ with arbitrarily large m: consider the values

$$n = 3s + 3,\ p = 1,\ m = 3s - 1,$$

for any integer $s \geq 1$. Conditions (1), (2) are fulfilled, hence the strings $a^{3s+3+1}(bbabb)^{3s-1}bbdc$ are in $L_{t,s}(\gamma) \cap R$ for all $s \geq 1$. This means that $L_{t,s}(\gamma) \cap R$ (hence $L_{t,s}(\gamma)$, too) is not a regular language: by pumping a substring of the suffix $(bbabb)^{3s-1}bbdc$ we get strings not in $L_{t,s}(\gamma) \cap R$.

Because the occurrences symbols c, d are introduced by the terminal rules of P_1, P_2, respectively, it follows that the strings in R are obtained by terminal derivations, that is, $L_{t,s}(\gamma) \cap R = L_{nt,s}(\gamma) \cap R$. Therefore, $L_{nt,s}(\gamma) \cap R \notin REG$, which implies that $L_{nt,s}(\gamma) \notin REG$ either. □

The synchronization is essential in the result above:

Theorem 4. $TNSL(RL) \subseteq REG, NTNSL(RL) \subseteq REG$.

Proof. Consider the layered grammar $\gamma = (N, T, t, (S_1, P_1), (S_2, P_2))$ and examine the derivations performed in the terminal non-synchronized mode. In each layer there is one nonterminal only, which moves from left to right. If the lower nonterminal is behind the upper one, then the derivation in the lower level depends on the transparent symbols in the upper layer (the lower level is blocked if its nonterminal is placed under a terminal symbol in the upper level). If the lower nonterminal goes ahead the upper one, then no restriction on the derivation is imposed. The two nonterminals can work freely, but the result is obtained by superposition. Because of the non-synchronization, we can apply a rule in any of the two layers. Therefore, (1) *only the relative position of the two nonterminals is important*, and (2) *as long as both nonterminals are present we can keep them at a bounded distance.* The distance between the two nonterminals can be bounded by $2m$, where

$$m = \max\{|u| \mid A \to u \in P_1 \cup P_2\}.$$

(If the distance between the two nonterminals is smaller than m, then we rewrite the nonterminal which is ahead; if the distance is between m and $2m$, then we rewrite the nonterminal which is behind. Note that we cannot bound this distance by m by always rewriting the nonterminal which is behind: if the upper nonterminal is behind, by rewriting it we can cover the lower nonterminal, which can modify the language, because the derivation is not synchonized. By first rewriting the lower nonterminal, we go at a distance at most $2m$, and the lower nonterminal continues to be observable after one more rule used in the upper layer.)

Therefore, the work of γ can be controlled by a "window" of length at most $2m$. Initially, this is $[S_1, S_2]$, it possibly grows to some $[w_1A_1, A_2]$ or $[A_1, w_2A_2]$, with $|w_1|, |w_2|$, terminal strings of length at most $2m-1$, and continues in this way until ending the derivation in one component; after that, only one nonterminal is enough in order to control the derivation.

We construct a right-linear grammar

$$G = (N', T \cup \{t\}, (S_1, S_2), P)$$

with

$$\begin{aligned} N' = \{&(w_1, w_2) \mid w_1 \in T^*(N \cup \{\lambda\}), w_2 \in T^*(N \cup \{\lambda\}), \\ &0 \leq |w_1|, |w_2| \leq 2m, \text{ at least one of } w_1, w_2 \text{ contains a nonterminal} \\ &\text{and at most one of them contains terminal symbols}\} \end{aligned}$$

and with the following rules.

We distinguish several cases, according to the type (terminal or nonterminal) of the strings in the two layers and to the length of these strings.

1. Both layers contain a nonterminal symbol and these symbols are superposed, that is, the sentential form ends with (A, B) and only A can be rewritten.

   (a) If $A \to uC$ is in $P_1, u \in (T \cup \{t\})^*$, then

   $$(A, B) \to (uC, B)$$

   is a production in P.

   (b) If $A \to u$ is in $P_1, u \in (T \cup \{t\})^*$, then

   $$(A, B) \to (u, B)$$

   is a production in P.

2. Both layers contain a nonterminal and the nonterminal in the second layer is behind, that is, the sentential form ends with a nonterminal (uA, B). The string u must be of the form $u = tu'$ in order to be able to apply a rule in the second layer. Note that it is not necessary to rewrite in the upper layer before rewriting the lower string and getting a longer string in the lower layer (the rewriting in the upper layer does not depend on the contents of the lower layer). Thus, we do not consider rules for modifying the symbol A in (uA, B).

(a) If $B \to vD$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(uA, B) \to u \diamond v_1(A, v_2D),$$

for $v = v_1v_2$ with $|u| = |v_1|$, is a production in P.

(b) If $B \to vD$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(uA, B) \to u_1 \diamond v(u_2A, D),$$

for $u = u_1u_2$ with $|u_1| = |v|$, is a production in P.

(c) If $B \to v$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(uA, B) \to u \diamond v_1(A, v_2),$$

for $v = v_1v_2$ with $|u| = |v_1|$, is a production in P.

(d) If $B \to v$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(uA, B) \to u_1 \diamond v(u_2A, \lambda),$$

for $u = u_1u_2$ with $|u_1| = |v|$, is a production in P.

3. Both layers contain a nonterminal and the nonterminal in the first layer is behind, that is, the sentential form ends with a nonterminal (A, vB), with $v \in (T \cup \{t\})^+$. (This time we need rules for modifying both symbols A and B – see again the mode of obtaining the bound $2m$ as a maximal distance between nonterminals in the two layers.)

(a) If $A \to uC$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(A, vB) \to u \diamond v_1(C, v_2B),$$

for $v = v_1v_2$ with $|u| = |v_1|$, is a production in P.

(b) If $A \to uC$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(A, vB) \to u_1 \diamond v(u_2C, B),$$

for $u = u_1u_2$ with $|u_1| = |v|$, is a production in P.

(c) If $B \to wD$ is in P_2, and $|vwD| \leq 2m$, then

$$(A, vB) \to (A, vwD)$$

is a production in P.

(d) If $A \to u$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(A, vB) \to u \diamond v(\lambda, B)$$

is a production in P.

(e) If $A \to u$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(A, vB) \to u_1 \diamond v(u_2, B),$$

for $u = u_1 u_2$ with $|u_1| = |v|$, is a production in P.

(f) If $B \to w$ is in P_2 and $|vw| \leq 2m$, then

$$(A, vB) \to (A, vw)$$

is a production in P.

4. Only the first layer contains a nonterminal, that is, the sentential form ends with a nonterminal (A, v).

(a) If $A \to uC$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(A, v) \to u \diamond v_1(C, v_2),$$

for $v = v_1 v_2$ with $|u| = |v_1|$, is a production in P.

(b) If $A \to uC$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(A, v) \to u \diamond v(C, \lambda)$$

is a production in P.

(c) If $A \to u$ is in P_1 and $u \in (T \cup \{t\})^*$, then

$$(A, v) \to u \diamond v$$

is a production in P.

5. Only the second layer contains a nonterminal, that is, the sentential form ends with a nonterminal (u, B); the string u must be of the form $u = tu'$ or $u = \lambda$ in order to be able to apply a rule in the second layer.

(a) If $B \to vD$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(u, B) \to u \diamond v(\lambda, D)$$

is a production in P.

(b) If $B \to vD$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(u, B) \to u_1 \diamond v(u_2, D),$$

for $u = u_1 u_2$ with $|u_1| = |v|$, is a production in P.

(c) If $B \to v$ is in P_2 and $v \in (T \cup \{t\})^*$, then

$$(u, B) \to u \diamond v$$

is a production of G.

This completes the construction.

One can easily check that we obtain $L(G) = L_{t,ns}(\gamma)$, which proves the inclusion $TNSL(RL) \subseteq REG$.

For the non-terminal case we can finish the derivation with the lower nonterminal placed under a symbol different from t. The necessary modifications are left to the reader. □

Theorems 3 cannot be improved by replacing RL by REG: even synchronized, layered grammars with regular components can generate only regular languages.

Theorem 5. *$TSL(REG) \subseteq REG, NTSL(REG) \subseteq REG$.*

Proof. Let us consider a layered grammar $\gamma = (N, T, t, (S_1, P_1), (S_2, P_2))$ with the sets P_1, P_2 containing regular rules. Because the work of γ is synchronized, whenever the nonterminal in the lower level is observable, it has to be rewritten. At the first step, when the upper level uses a rule different from $A \to tB$,$A \to t$ (that is a terminal in T is introduced), this will cover the nonterminal in the lower level, hence it is no longer rewritten. Thus, the derivation of γ starts by a number of steps of using rules of the form $A \to tB$ (this number may be zero). Synchronously, the lower level nonterminal advances, one step behind the nonterminal in the upper level, at any step it can be replaced by a terminal, and this ends the derivation in the lower level. When the upper level introduced a symbol in T, the lower one should use a terminal rule in the case of terminal derivation, or any rule in the case of non-terminal derivation and it stops. The upper level can continue without any restriction. To sum up, a window of length two suffices in order to control the work of γ in a way similar to that in the proof of Theorem 4. Consequently, $L_{t,s}(\gamma) \in REG, L_{nt,s}(\gamma) \in REG$. □

Corollary 5. *$YL(REG) = REG$ for all $Y \in \{TS, TNS, NTS, NTNS\}$.*

Proof. Combine Lemma 5 (it gives the inclusion $\supseteq$) with Theorem 4 (the converse inclusion for the non-synchronized case) and Theorem 5 (the converse inclusion for the synchronized case). □

Thus, the regular layered grammars need no further investigation (in what concerns the generative capacity). The results above deserve to be emphasized: in the synchronized case, the regular rules are strictly less powerful than right-linear rules. This does not happen in many situations in formal language theory (for instance, in regulated rewriting area, [6]). However, a similar result has been recently proved for parallel communication grammar systems: in the centralized returning case, the regular rules are strictly weaker than the right-linear rules [7].

7 A Characterization of RE Languages

We now increase the degree of synchronization in a layered grammar, by specifying the pair of rules to be used by the two components, at steps when both of them have observable nonterminals.

A *matrix layered grammar* is a construct

$$\gamma = (N, T, t, S_1, S_2, M),$$

where N, T, t, S_1, S_2 are as in a usual layered grammar (the nonterminal and the terminal alphabet, the transparent symbol, the axioms of the two layers, respectively), and M is a finite set of pairs (we call them *matrices*) of the forms

$$(A_1 \to u_1, A_2 \to u_2),\ (A_1 \to u_1, \#),\ (\#, A_2 \to u_2),$$

where $A_1 \to u_1, A_2 \to u_2$ are context-free rules over $N \cup T \cup \{t\}$. In a derivation step $[x_1, x_2] \to [y_1, y_2]$ we have to use a pair $(A_1 \to u_1, A_2 \to u_2)$ if both x_1 and x_2 contain observable nonterminals (hence $x_1 = x_1' A_1 x_1'', y_1 = x_1' u_1 x_1'', x_2 = x_2' A_2 x_2'', y_2 = x_2' u_2 x_2''$), a pair $(A_1 \to u_1, \#)$ if x_2 contains no observable nonterminal and A_1 appears in x_1 (hence $x_1 = x_1' A_1 x_1'', y_1 = x_1' u_1 x_1'', x_2 = y_2$), and a pair $(\#, A_2 \to u_2)$ if x_1 is a terminal string and A_2 is an observable nonterminal in x_2 (hence $x_1 = y_1, x_2 = x_2' A_2 x_2'', y_2 = x_2' u_2 x_2''$). We denote by $L_t(\gamma)$ the language generated by γ in the terminal mode and by $L_{nt}(\gamma)$ the language generated by γ in the non-terminal mode. The corresponding families of languages are denoted by $TML(X), NTML(X), X \in \{REG, RL, LIN, CF\}$.

Remark. Because we may always assume that the nonterminals used in the two layers are distinct, we can consider matrices as pairs of rules without specifying where each rule is used: we have just to use the two rules in two different layers. We prefer here to work with the previous definition because we find it more natural.

Rather surprisingly, the following characterization of recursively enumerable languages can be obtained.

Theorem 6. *For every language $L \in RE$ there are a projection h, a regular language R, and a language $L' \in TML(RL) \cap NTML(RL)$ such that $L = h(L' \cap R)$.*

Proof. We use the following variant (proved in [8]) of the characterization of recursively enumerable languages by means of equality sets of morphisms (see [5], [12], [13]). For two morphisms $h_1, h_2 : V^* \longrightarrow U^*$ we denote

$$EQ(h_1, h_2) = \{w \in V^* \mid h_1(w) = h_2(w)\}.$$

For every language $L \in RE, L \subseteq V^*$, there are two alphabets V_1, V_2 such that $V \subseteq V_2$, two λ-free morphisms $h_1, h_2 : V_1^* \longrightarrow V_2^*$, a regular set $R \subseteq V_2^*$, and a projection $h_3 : V_2^* \longrightarrow V^*$, such that $L = h_3(h_1(EQ(h_1, h_2)) \cap R)$. Consider also the alphabet of new symbols $V_2' = \{a' \mid a \in V_2\}$. We now construct the matrix layered grammar

$$\gamma = (\{S_1, S_1', S_2\}, V_2 \cup V_2' \cup \{c, d\}, t, S_1, S_2, M),$$

with the following matrices of rules:

(1) $(S_1 \to tS_1', \#)$,

(2) $(S_1' \to tb_{i_1}tb_{i_2}\dots tb_{i_k}tS_1',\ S_2 \to b'_{j_1}tb'_{j_2}t\dots tb'_{j_l}tS_2)$,

(3) $(S_1' \to tb_{i_1}tb_{i_2}\dots tb_{i_k}td,\ S_2 \to b'_{j_1}tb'_{j_2}t\dots tb'_{j_l}tc)$,
for $h_1(a) = b_{i_1}b_{i_2}\dots b_{i_k}, k \geq 1, b_{i_s} \in V_2, 1 \leq s \leq k$,
and $h_2(a) = b'_{j_1}b'_{j_2}\dots b'_{j_l}, l \geq 1, b'_{j_s} \in V_2', 1 \leq s \leq l$.

Consider also the regular language

$$R_0 = \{b'b \mid b \in V_2\}^*\{cd\}$$

and the projection $g : (V_2 \cup V_2')^* \to V_2^*$ defined by $g(b) = b, b \in V_2$, and $g(b') = \lambda, b' \in V_2'$. It is easy to see that we have

$$g(L_t(\gamma) \cap R_0) = g(L_{nt}(\gamma) \cap R_0) = h_1(EQ(h_1, h_2)).$$

Indeed, because the upper component introduces the symbol t in each odd position and the nonterminal of the lower component appears always in an odd position, the non-terminal derivations should be terminal. Moreover, the intersection with R_0 ensures the fact that we end in the two layers with two identical strings modulo the primes appearing in the lower layer, namely the two strings correspond to some $h_1(w_1) = h_2(w_2)$; the matrices in M ensure the fact that $w_1 = w_2$: after using the only matrix of type (1), always is S_2 observable, and the two components should stop at the same time, by using a matrix of type (3).

Now, let us consider the morphism $h : V_2^* \longrightarrow (V_2 \cup V_2')^*$ defined by $h(b) = b'b, b \in V_2$, the regular language $R' = h(R)cd$, and extend the projection h_3 to $h_3' : (V_2 \cup V_2')^* \to V^*$ by $h_3'(b) = h_3(b), b \in V_2$, and $h_3'(b') = \lambda, b' \in V_2'$, $h_3'(c) = h_3'(d) = \lambda$. Then we have

$$L = h_3'(L_t(\gamma) \cap h(R')) = h_3'(L_{nt}(\gamma) \cap h(R')).$$

Indeed, R' plays at the same time the roles of both R and R_0, while h_3' plays at the same time the roles of h_3 and g. □

Corollary 6. *For every family of languages $F \subset RE$ which is closed under intersection with regular languages and projections we have $TML(RL) - F \neq \emptyset$, $NTML(RL) - F \neq \emptyset$.*

Proof. An inclusion $TML(RL) \subseteq F, NTML(RL) \subseteq F$ would imply that the closure of $TML(RL), NTML(RL)$ under intersection with regular languages and projections is included into the closure of F under these operations. This implies the inclusion $RE \subseteq F$, contradicting the strict inclusion $F \subset RE$. □

Important families F as above are MAT^{λ}, of languages generated by matrix grammars with arbitrary context-free rules but without appearance checking, and $ET0L$, the family of languages generated by extended tabled interactionless Lindenmayer systems, [10]. The proof of Theorem 5 remains valid also for matrix layered grammars, that is the next result holds:

Theorem 7. $TML(REG) = NTML(REG) = REG$.

8 Final Remarks; Variants

Several variants of layered grammars can be naturally defined. We only mention some of them as a proof of the richness of this notion. First, as it is the case with the icons observable in windows superposed on the computer screen, we can assume that some nonterminals are "more active" than others. When several nonterminals are observable, the "most active" of them is rewritten. This idea can be implemented, for instance, under the form of a partial order relation on the nonterminal alphabet of the grammar, or under the form of a leftmost restriction on the derivation (the first observable nonterminal from the left of the layer should be rewritten). Note that in the case of linear grammars (hence also of right-linear and regular grammars) these variants coincide with the one investigated here, hence all Theorems 1 – 7 from the previous sections remain true also for these variants. Second, we can consider a variant which removes the apparent contrast between the parallel character of activating symbols by observability and the sequential mode of rewriting. That is, it is quite natural to derive all observable symbols at the same time. This leads to considering parallel derivations, either in a context-free grammar as here (this reminds the so-called Indian and Russian parallel grammars in regulated rewriting area, see [6]), or in a pure grammar, where there is no distinction between terminal and nonterminal symbols (this corresponds to Lindenmayer systems; in particular, layered 0L or D0L systems look attractive). Third, we can consider layered grammars of any order, not only with two levels as we have done here. The definition of observability can be obviously extended to n-layered strings; similarly the definition of a layered grammar of degree $n, n \geq 1$, is an obvious extension of the definition here. Such systems will probably have a rather intricate behavior. They correspond in a better way to a parallel grammar system, with a particular type of cooperation among components: they work synchronously, on their separate sentential forms (this is similar to a parallel communication grammar system, [9], [4]), but they do not communicate by sending messages, rather they just influence each other through observable symbols; however, the result is highly integrated: it is the superposition (in the sense of the operation $\diamond$, extended to n-layered strings) of the strings generated by the component grammars. Then, usual derivation, leftmost, parallel derivations can be considered also in this case. Of course, such a system with n layers can be simulated by a system with $n + 1$ layers (just add a component which contains only the rule $S \to t$). Whether or not the systems of degree $n + 1$ are strictly more powerful than those of degree n becomes a fundamental problem (in general, not solved in grammar systems area, [4]). By the various motivations of the model, by the results given here (especially the possibility of generating non-regular languages by layered grammars with right-linear rules and the characterization of recursively enumerable languages by matrix layered grammars with rules of the same type — right-linear), and by the wealth of problem raised by the variants mentioned above, the layered grammars prove to be a research area of definite interest. Of course, it is however premature to inquiry about the practical relevance of these grammars.

Acknowledgements. Gh. Păun was supported by a grant under the program for visiting professors of the Group for Mathematical Informatics (GNIM) of the Italian National Research Council (CNR).

Thanks are due to an anonymous referee, who made many comments on an earlier version of the paper; the structuring of the construction in the proof of Theorem 4 is also due to him/her.

References

[1] M. Andraşiu, J. Dassow, Gh. Păun, A. Salomaa, Language-theoretic problems arising from Richelieu cryptosystems, *Theor. Comput. Sci.*, 116 (1993), 339 – 357.

[2] P. Bottoni, M. F. Costabile, S. Levialdi, P. Mussio, Defining visual languages for interactive computing, *IEEE Transactions on Systems, Man, and Cybernetics – A*, 27 (1997), 773 – 782.

[3] P. Bottoni, M. F. Costabile, S. Levialdi, P. Mussio, Specification of Visual Languages as Means for Interaction, *Theory of Visual Languages*, K. Marriott, B. Meyer eds., Springer-Verlag, 1997, to appear.

[4] E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

[5] K. Culik II, A purely homomorphic characterization of recursively enumerable sets, *Journal of the ACM*, 26 (1979), 345 – 350.

[6] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, Heidelberg, 1989.

[7] S. Dumitrescu, Gh. Păun, On the power of parallel communicating grammar systems with right-linear components, *Rev. Fr. Aut. Inform. Theor., RAIRO, Theor. Informatics*, 31, 4 (1997), 331 – 354.

[8] L. Kari, Gh. Păun, G. Rozenberg, A. Salomaa, S. Yu, DNA computing, sticker systems, and universality, *Acta Informatica*, 35 (1998), 401 – 420.

[9] Gh. Păun, L. Sântean, Parallel communicating grammar systems: the regular case, *Ann. Univ. Buc., Matem.-Inform. Series*, 38, 2 (1989), 55 – 63.

[10] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, 1980.

[11] G. Rozenberg, A. Salomaa, Eds., *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Heidelberg, 1997.

[12] A. Salomaa, Equality sets for homomorphisms of free monoids, *Acta Cybernetica*, 4 (1978) 127 – 139.

[13] A. Salomaa, *Jewels of Formal Language Theory*, Computer Science Press, Rockville, Maryland, 1981.

Received October, 1997

Acta Cybernetica **13** (1998) 359-373.

On extended simple eco-grammar systems *

Judit CSIMA†

Abstract

In this contribution extended simple eco-grammar systems are studied. A simple eco-grammar system is formed from an environment given by a set of 0L rules and from some agents represented by sets of CF rules. In an extended simple eco-grammar system we distinguish a subset of the alphabet of the system and only strings over this subalphabet are in the generated language. The relations between language classes generated with different parameters and derivation modes are investigated.

1 Introduction

The concept of the eco-grammar system (the EG system, for short) has been introduced in [2] as a model of communities of agents which interact with their common shared environment. Several aspects of these systems were discussed in [3] and [6], properties of a restricted variant, called simple eco-grammar system, were studied in [4], [1], [10], [5], and [9]. Briefly, a simple eco-grammar system consists of several agents (represented by sets of context-free rules) and an environment (given by a set of $0L$ rules). At any moment of time, the behaviour of the system is described by the state of the environment which is a string over the alphabet of the system. The environmental state changes by derivation steps. In a derivation step the agents act on the string by applying one of their context-free rules - each agent rewrites only one letter - and the environment replaces, according to its $0L$ rule set, in a parallel manner the symbols where the agents do not perform any action.

Starting from an initial string representing the environment, a lot of strings following each other arise, which describe the evolving system. The language generated by the eco-grammar system is the set of strings which can be obtained from the initial environmental state by a sequence of derivation steps. In the case of extended simple eco-grammar systems only those strings belong to the determined language which are over a distinguished subset of the alphabet of the system, the terminal alphabet. This notion was introduced in [5] under the name of 0-terminal EG system and some basic properties of these systems were examined as well. It

*Research supported by the Hungarian Scientific Foundation "OTKA" Grant No. T017105. Presented at the Conference of PhD Students on Computer Science, July 18-22 1998, Szeged.

†Computer and Automation Research Institute, Hungarian Academy of Sciences, Kende u. 13-17, H-1111, Budapest, Hungary. E-mail: csima@luna.aszi.sztaki.hu

was shown there that λ-free extended simple eco-grammar systems with one agent are as powerful as λ-free extended simple EG systems with at most n agents, if the original mode of the derivation is used.

Following this line, in this contribution we deal with a more sophisticated version of the derivation, the team behaviour of the extended simple eco-grammar systems: in each derivation step from the n agents exactly k or at most k are allowed to work. We describe the behaviour and the generative power of these systems according to some size parameters: the total number of the agents, the number of the agents being active in a derivation step. We examine the hierarchy of the language classes generated by extended simple eco-grammar systems with and without λ-rules.

The results demonstrate that while in the non-extended case the size parameters of the teams and the agent population have influence on the power of the system, in the extended case these parameters are not important: we obtain a collapsing hierarchy.

2 Preliminaries and the definition of the extended simple eco-grammar system

In this section we present the basic notions and notations used in the paper, for further information the reader is referred to [8] and [7].

An alphabet is a non-empty set of symbols. The set of all non-empty words over a finite alphabet V is denoted by V^+, the empty word is denoted by λ. The set V^* is $V^+ \cup \{\lambda\}$.

By a context free production or by a context free rule (a CF rule, for short) over an alphabet V we mean a production of the form $a \to u$, where $a \in V$ and $u \in V^*$. A CF rule is a λ-rule (or an erasing rule) if $u = \lambda$.

A $0L$ system is a construct $H = (V, P, \omega)$, where V is a finite alphabet, P is a finite set of context free rules over V and $\omega \in V^*$ is the axiom. Moreover, P has to be complete, that is for each symbol a from V there must be at least one rule in P with this letter on the left-hand side.

$0L$ systems use parallel derivations: we say that x directly derives y in a $0L$ system $H = (V, P, \omega)$, written as $x \Rightarrow_H y$, if $x = x_1 x_2 \dots x_n$, $y = y_1 y_2 \dots y_n$, $x_i \in V$, $y_i \in V^*$ and the rules $x_i \to y_i$ are in P for $1 \leq i \leq n$.

The generated language of a $0L$ system H (denoted by $L(H)$) is the set of the words over V which can be derived (in some steps) from the axiom.

Throughout the paper, we use the customary notations: $\subseteq$ denotes inclusion and $\subset$ denotes strict inclusion.

If L is a language, we call $alphL$ the set of all letters which occur in the words of L.

If V is an alphabet, we will use the following notations:

- $V^{(k)} = \{ A^{(k)} \mid A \in V \}$, where k is a positive integer,
- $V' = \{ A' \mid A \in V \}$,

- $\overline{V} = \{ \overline{A} \mid A \in V \}$.

If u is a word of the form $u = x_1 \dots x_n$, $x_i \in V$, $1 \leq i \leq n$, then $u^{(k)} = x_1{}^{(k)} \dots x_n{}^{(k)}$ and $u' = x_1' \dots x_n'$.

After these basic notions we present the definition of the extended simple eco-grammar system.

Definition 2.1
An extended simple eco-grammar system is a construct $\Sigma = (V_E, P_E, R_1, \dots, R_n, \omega, \Delta)$, *where*

- V_E *is a finite alphabet,*
- P_E *is a finite set of CF rules over* V_E, *this set is complete i.e. for each letter of* V_E *there exists at least one rule in* P_E *with this letter on the left-hand side,*
- R_i *is a finite, non-empty set of CF rules over* V_E *for* $1 \leq i \leq n$,
- $\omega \in V_E{}^*$,
- Δ *is a non-empty subset of* V_E.

In this construct V_E is the alphabet and P_E is the set of the evolution rules of the environment. R_i represents the ith agent, this is the set of its action rules. The current state of the environment, which is also the state of the eco-grammar system, is the current sentential form. String ω is the initial state from which all the derivations start. Δ is the terminal set, the language of the EG system consists of words over Δ.

The system changes its state by a simultaneous action of some agents and by a parallel rewriting according to P_E. In this contribution we consider two types of derivations, first we present the definition of derivation mode $= k$.

Definition 2.2
Consider an extended simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \dots, R_n, \omega, \Delta)$. *We say that* x *directly derives* y *in* Σ *in mode* $= k$ ($x \in V_E{}^+, y \in V_E{}^*$ *and* $1 \leq k \leq n$), *written as* $x \overset{=k}{\Longrightarrow}_\Sigma y$, *if*

- $x = x_1 Z_1 x_2 Z_2 \dots x_k Z_k x_{k+1}$, $Z_i \in V_E$, $x_j \in V_E{}^*$,
- $y = y_1 w_1 y_2 w_2 \dots y_k w_k y_{k+1}$, $y_i, w_j \in V_E{}^*$,
- *there exist* k *different agents in* Σ, *namely* $R_{j_1}, R_{j_2}, \dots, R_{j_k}$, *such that* $Z_i \to w_i \in R_{j_i}$ *for* $1 \leq i \leq k$, *and*
- $x_i = y_i = \lambda$ *or* $x_i \Rightarrow_E y_i$, *where* $E = (V_E, P_E, \omega)$ *is the* $0L$ *system of the environment.*

In the above case we say that $x \overset{=k}{\Longrightarrow}_\Sigma y$ is a derivation step.

Another mode of the derivation is derivation mode $\leq k$:

Definition 2.3
Let $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega, \Delta)$ be an extended simple eco-grammar system. We say that x directly derives y in Σ by a derivation $\leq k$ $(1 \leq k \leq n)$, written as $x \overset{\leq k}{\Longrightarrow}_\Sigma y$, if $x \overset{=l}{\Longrightarrow}_\Sigma y$ for some l, $1 \leq l \leq k$.

We denote the transitive and reflexive closure of $\overset{=k}{\Longrightarrow}_\Sigma$ and $\overset{\leq k}{\Longrightarrow}_\Sigma$ by $\overset{=k}{\Longrightarrow}^*_\Sigma$ and $\overset{\leq k}{\Longrightarrow}^*_\Sigma$. In both cases, the generated language consists of the words which can be generated from the axiom in some derivation steps and which are over Δ.

Definition 2.4
Consider an extended simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega, \Delta)$. The generated languages in mode $= k$ and $\leq k$ are the following:

$$L(\Sigma, = k) = \{ v \in \Delta^* \mid \omega \overset{=k}{\Longrightarrow}^*_\Sigma v \},$$

$$L(\Sigma, \leq k) = \{ v \in \Delta^* \mid \omega \overset{\leq k}{\Longrightarrow}^*_\Sigma v \}.$$

Now we present an example.

Example 2.1 *Let $\Sigma = (V_E, P_E, R_1, R_2, R_3, \omega, \Delta)$ be the following extended simple EG system:*

- $V_E = \{ A, B, C, a, b, c \}$,
- $P_E = \{ A \to a, B \to b, C \to c, a \to a, b \to b, c \to c \}$,
- $R_1 = \{ A \to A^2, A \to a \}$,
- $R_2 = \{ B \to B^2, B \to b \}$,
- $R_3 = \{ C \to C^2, C \to c \}$,
- $\omega = ABC$,
- $\Delta = \{ a, b, c \}$.

It is easy to see that in mode $= 3$ a derivation sequence is of the following form. (In order to simplify the writing, when a rule can be applied in several places we use the left-most one; the other derivations give the same word.)

$$ABC \Rightarrow A^2B^2C^2 \Rightarrow A^2aB^2bC^2c \Rightarrow \ldots \Rightarrow A^2a^{k-2}B^2b^{k-2}C^2c^{k-2} \Rightarrow a^kb^kc^k$$

The three agents have to finish the derivation at the same step, otherwise the derivation would be blocked. (All of the three agents have to be active in each step, if there are at most two different upper case letters in the sentential form the derivation is blocked.)

Thus the generated language is $L(\Sigma, = 3) = \{ a^n b^n c^n \mid n \geq 1 \}$.

Finally, we present some notations which we will use in the paper.
We consider extended simple eco-grammar systems with or without λ rules. When we allow λ-rules in P_E and in the sets R_i, we use the following notations: the class of the languages which can be generated by an extended simple eco-grammar system is denoted by $\mathcal{L}(EE)^\lambda$ (in this notation the first E means "extended", the second one refers to "eco"). The class of the languages generated by a system containing n agents and operating in mode $= k$ ($\leq k$) is denoted by $\mathcal{L}(EE^\lambda(n, = k))$ ($\mathcal{L}(EE^\lambda(n, \leq k))$).
We omit the notation λ if λ-rules are not allowed neither in P_E nor in the sets R_i: $\mathcal{L}(EE)$, $\mathcal{L}(EE(n, = k))$ and $\mathcal{L}(EE(n, \leq k))$. In the statements of the paper we use the notation (λ) if a statement is true both with and without λ-rules.

3 Relations between the language classes generated by extended simple EG systems

3.1 The hierarchy of language classes $\mathcal{L}(EE^{(\lambda)}(n, = k))$

In this section we are going to present results about the hierarchy of the language classes generated by n agents in derivation mode $= k$. This question was examined for the non-extended simple eco-grammar systems in [4]. It was proved there that the generated language classes are incomparable in most of the cases. We get very different results for the extended systems: here the language classes form a collapsing hierarchy.

Most of the statements and proofs of the section are true with or without λ-rules, this fact is denoted by the superscript (λ); sometimes a statement is true in both cases, but the proofs are different, in this case we will present the two different proofs together; and sometimes a statement is true only when λ-rules are allowed, in this case we will emhasize this fact.

First we examine the role of the first parameter, the number of the agents.

Lemma 3.1.1 *For* $1 \leq k \leq n \leq m$, $\mathcal{L}(EE^{(\lambda)}(n, = k)) \subseteq \mathcal{L}(EE^{(\lambda)}(m, = k))$.

Proof
We show that for any $1 \leq k \leq n \leq m$ and for any extended simple EG system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega, \Delta)$ there exists another extended simple EG system $\Sigma' = (V_E', P_E', R_1', \ldots, R_m', \omega', \Delta')$ such that $L(\Sigma, = k) = L(\Sigma', = k)$. Moreover, if Σ does not contain λ-rules then Σ' is also λ-free.
Let Σ' be the following:

- $V_E' = V_E \cup \{D\}$, where $D \notin V_E$,
- $P_E' = P_E \cup \{D \to D\}$,
- $R_i' = R_i$ for $1 \leq i \leq n$,

- $R_i' = \{D \to D\}$ for $n+1 \leq i \leq m$,
- $\omega' = \omega$,
- $\Delta' = \Delta$.

It is clear that $L(\Sigma, = k) = L(\Sigma', = k)$ and Σ' is λ-free if and only if Σ does not contain λ-rules. ■

The following result shows that the above inclusion is not proper. The statement is true with or without λ-rules.

Lemma 3.1.2 *For* $1 \leq k \leq n$, $\mathcal{L}(EE^{(\lambda)}(n+1, = k)) \subseteq \mathcal{L}(EE^{(\lambda)}(n, = k))$.

Proof
The λ-free case
We will show that we can simulate any λ-free extended simple EG system $\Sigma = (V_E, P_E, R_1, \ldots, R_{n+1}, \omega, \Delta)$ working in mode $= k$ by another λ-free extended simple EG system $\Sigma' = (V_E', P_E', R_1', \ldots, R_n', \omega', \Delta')$ working also in mode $= k$. Let Σ' be the following system:

- $V_E' = V_E^{(1)} \cup V_E^{(2)} \cup V_E' \cup \{D\}$, where $D \notin V_E$,
- $P_E' = P_E^{(1)\to(2)} \cup \{ A^{(2)} \to A^{(1)} \mid A \in V_E \} \cup \{ A' \to D \mid A \in V_E \} \cup$
  $\cup \{ D \to D \}$, where $P_E^{(1)\to(2)} = \{ A^{(1)} \to v^{(2)} \mid A \to v \in P_E \}$,
- $R_1' = R_1^{(1)\to(2)} \cup R_{n+1}'^{(1)\to(2)} \cup \{A' \to A^{(1)} \mid A \in V_E \} \cup$
  $\cup \{A^{(2)} \to A^{(1)} \mid A \in V_E \}$,
- $R_i' = R_i^{(1)\to(2)} \cup R_{n+1}'^{(1)\to(2)} \cup \{A^{(2)} \to A^{(1)} \mid A \in V_E \}$ for $2 \leq i \leq n$,
  where
  $R_j^{(1)\to(2)} = \{ A^{(1)} \to w^{(2)} \mid A \to w \in R_j \}$ for $1 \leq j \leq n$
  and
  $R_{n+1}'^{(1)\to(2)} = \{ v^{(1)} \to w'^{(2)} \mid v \to w \in R_{n+1} \}$, where
  $w'^{(2)} = x_1^{(2)} \ldots x_{m-1}^{(2)} x_m'$ if $w = x_1 \ldots x_{m-1} x_m$, $x_i \in V_E$, and $1 \leq i \leq m$,
- $\omega' = \omega^{(1)}$,
- $\Delta' = \Delta^{(1)}$.

The main idea of this construction is the following. We simulate one derivation step of Σ by two derivation steps of Σ'. In the first simulation step we use the rules corresponding to the derivation of Σ, the second step checks if the simulation is correct.
In Σ' the set R_{n+1} is included in the set of rules of all agents in form of $R_{n+1}'^{(1)\to(2)}$. When an agent uses a rule corresponding to R_{n+1}, a primed letter is introduced as well. The only agent which is able to make these primed letters disappear is R_1'; this construction guarantees (considering the rules of P_E') that the derivation

cannot terminate with a terminal word if the agents use more than one rule from ${R'}_{n+1}^{(1)\to(2)}$ at the same derivation step.

If in a derivation step of Σ the agents $R_{i_1}, \ldots, R_{i_k}$ work and $i_j \leq n$ for $1 \leq j \leq k$, the simulation goes as follows.

In the first simulation step in Σ', the agents ${R_{i_1}}', \ldots, {R_{i_k}}'$ work using rules from ${R_{i_j}}^{(1)\to(2)}$, corresponding to the step of Σ. Then the environment uses the same rules as in Σ (of the form ${P_E}^{(1)\to(2)}$).

In the second simulation step the agents ${R_{i_1}}', \ldots, {R_{i_k}}'$ rewrite k letters and the environment rewrites the remaining letters by using the rules of the form $A^{(2)}\to A^{(1)}$.

If R_{n+1} was among the agents $R_{i_1}, \ldots, R_{k-1}, R_k = R_{n+1}$ in a derivation step in Σ, the two simulation steps of Σ' are the following.

In the first simulation step we choose one agent ${R_s}'$ in Σ' such that $s \neq i_j$ for any $1 \leq j \leq k-1$. It is possible because $k \leq n$. This agent will simulate the work of R_{n+1} by using the corresponding rule of ${R'}_{n+1}^{(1)\to(2)}$.

The agents ${R_{i_1}}', \ldots, {R'}_{k-1}$ simulate the remaining rules of the agents and the environment rewrites the remaining letters in the same way as in Σ.

In the second simulation step ${R_1}'$ rewrites the primed letter, other $k-1$ agents rewrite $k-1$ other letters and the environment rewrites the remaining letters into symbols of ${V_E}^{(1)}$.

Since we can simulate every derivation step of Σ by a derivation sequence of Σ', we obtain that $L(\Sigma, = k) \subseteq L(\Sigma', = k)$.

On the other hand, we can simulate by Σ the derivation sequences of Σ' resulting terminal words, which gives the other inclusion $L(\Sigma', = k) \subseteq L(\Sigma, = k)$.

We will simulate two derivation steps of these sequences of Σ' by one step of Σ.

For each $v \in L(\Sigma', = k)$ (which implies $v \in {\Delta^{(1)}}^+$ because λ-rules are not allowed) there exists a derivation sequence in Σ' of the form

$$\omega' \Rightarrow_{\Sigma'} v_1 \Rightarrow_{\Sigma'} \ldots \Rightarrow_{\Sigma'} v_t = v.$$

Since $v \in {V_E}^{(1)^+}$, there are neither primed letters nor D in v.

Considering the possible rules in ${P_E}'$, it is easy to see that there can be at most one primed letter in the previous word v_{t-1}.

The fact that there is no primed letter in v_{t-1} means that in the $(t-1)th$ step k agents: ${R_{i_1}}', \ldots {R_{i_k}}'$ used rules from the sets ${R_{i_j}}^{(1)\to(2)}$ with $1 \leq i_j \leq n$. In this case we can simulate the last two derivation steps of Σ' by one derivation step of Σ, using the corresponding rules of the agents $R_{i_1}, \ldots R_{i_k}$ and the environment.

The other possibility gives that $k-1$ agents, ${R_{i_1}}', \ldots, {R_{i_{k-1}}}'$ use rules from the sets ${R_{i_j}}^{(1)\to(2)}$ with $1 \leq i_j \leq n$ and one agent uses a rule of ${R'}_{n+1}^{(1)\to(2)}$.

Then we can simulate the last two steps by one step in Σ when the agents $R_{i_1}, \ldots, R_{i_{k-1}}, R_{n+1}$ and the environment use the corresponding rules.

In both cases $v_{t-2} \in {V_E}^{(1)^+}$, thus we can continue the simulation by giving the role of v to v_{t-2}. Since the axiom is over ${V_E}^{(1)}$ and t is an even number, we can simulate the whole derivation sequence.

With λ-rules

The main idea is the same as in the λ-free case, we will show that we can simulate any extended simple EG system $\Sigma = (V_E, P_E, R_1, \ldots, R_{n+1}, \omega, \Delta)$ working in mode $= k$ by another extended simple EG system
$\Sigma' = (V_E', P_E', R_1', \ldots, R_n', \omega', \Delta')$ working also in mode $= k$.
Let Σ' be the following system:

- $V_E' = V_E^{(1)} \cup V_E^{(2)} \cup \{D, T_1, T_2\}$, where $D, T_1, T_2 \notin V_E$,
- $P_E' = P_E{}^{(1)\to(2)} \cup \{ A^{(2)} \to A^{(1)} \mid A \in V_E \} \cup \{ D \to D, T_1 \to D, T_2 \to \lambda \}$,
- $R_1' = R_1{}^{(1)\to(2)} T_2 \cup R_{n+1}^{(1)\to(2)} T_1 \cup \{ T_1 \to \lambda \} \cup \{ T_2 \to \lambda \}$,
- $R_i' = R_i{}^{(1)\to(2)} T_2 \cup R_{n+1}^{(1)\to(2)} T_1 \cup \{ T_2 \to \lambda \}$ for $2 \le i \le n$,
  where
  $R_j{}^{(1)\to(2)} T_2 = \{ A^{(1)} \to w^{(2)} T_2 \mid A \to w \in R_j \}$ for $1 \le j \le n$ and
  $R_{n+1}^{(1)\to(2)} T_1 = \{ A^{(1)} \to w^{(2)} T_1 \mid A \to w \in R_{n+1} \}$,
- $\omega' = \omega^{(1)}$,
- $\Delta' = \Delta^{(1)}$.

The proof that $L(\Sigma, = k) = L(\Sigma', = k)$ is very similar to that of the λ-free case, T_1 plays the role of the primed letters, it controls the usage of the rules of $R'^{(1)\to(2)}_{n+1}$, T_2 letters guarantee that k agents can be active in the second simulation step even if in the first simulation step λ-rules were applied. ■

We obtain from the two previous lemmas the following corollary, which means that the first parameter has no influence on the class of the generated languages.

Corollary 3.1.1 *For* $1 \le k \le n$, $\mathcal{L}(EE^{(\lambda)}(n, = k)) = \mathcal{L}(EE^{(\lambda)}(k, = k))$.

We have seen that there is not any hierarchy according to the number of agents. Now we turn our attention to the second parameter: the number of the agents working in a step. Considering the above corollary, it is enough to examine the relation of the language classes $\mathcal{L}(EE^{(\lambda)}(k, = k))$.

For the λ-free case, it was proved in [5] that $\mathcal{L}(EE(n+1, = n+1))$ is included in $\mathcal{L}(EE(n, = n))$. In the following lemma we present the same result for extended simple EG systems with λ-rules as well. We give a simulation which is based on the construction used in [5], thus in the first part of our proof we briefly summarize the construction and the explanation used there for the λ-free case. Then in the second part of the proof we give the modifications which are necessary to obtain the same result when λ-rules are allowed.

Lemma 3.1.3 *For* $1 \le n$, $\mathcal{L}(EE^{(\lambda)}(n+1, = n+1)) \subseteq \mathcal{L}(EE^{(\lambda)}(n, = n))$.

Proof
The λ-free case (from [5]):

For any extended simple EG system $\Sigma = (V_E, P_E, R_1, \ldots, R_{n+1}, \omega, \Delta)$ we give another extended simple EG system $\Sigma' = (V_E', P_E', R_1', \ldots, R_n', \omega', \Delta')$, such that $L(\Sigma, = n+1) = L(\Sigma', = n)$.
Let Σ' be the following:

- $V_E' = V_E \cup V_E' \cup \{\bar{A} \mid A \to \alpha \in R_{n+1}\} \cup \{D\}$, where $D \notin V_E$,
- $P_E' = \{ A \to \alpha' \mid A \to \alpha \in P_E \} \cup \{ A \to \bar{A} \mid A \to \alpha \in R_{n+1} \} \cup$
  $\cup \{ A' \to A \mid A \in V_E \} \cup \{ \bar{A} \to D \mid A \to \alpha \in R_{n+1} \} \cup \{ D \to D \}$,
- $R_1' = \{A \to \alpha' \mid A \to \alpha \in R_1 \} \cup \{\bar{A} \to \alpha \mid A \to \alpha \in R_{n+1} \}$,
- $R_i' = \{A \to \alpha' \mid A \to \alpha \in R_i \} \cup \{A' \to A \mid A \in V_E \}$ for $2 \leq i \leq n$,
- $\omega' = \omega$,
- $\Delta' = \Delta$.

Let the step

$$x = x_1 Z_1 x_2 Z_2 \ldots x_{n+1} Z_{n+1} x_{n+2} \Rightarrow y = y_1 w_1 y_2 w_2 \ldots y_{n+1} w_{n+1} y_{n+2}$$

be a derivation step in Σ, where the rules used by the agents are $Z_i \to w_i$ for $1 \leq i \leq n+1$ and the derivations $x_j \Rightarrow y_j$ for $1 \leq j \leq n+2$ are performed by the environment. We can simulate this step by two steps of Σ' in the following way. (We suppose that R_{n+1} used the rule $Z_{n+1} \to w_{n+1}$.)

$$x = x_1 Z_1 x_2 Z_2 \ldots x_{n+1} Z_{n+1} x_{n+2} \Rightarrow x' = y_1' w_1' y_2' w_2' \ldots y_{n+1}' \overline{Z}_{n+1} y_{n+2}' \Rightarrow$$

$$\Rightarrow y = y_1 w_1 y_2 w_2 \ldots y_{n+1} w_{n+1} y_{n+2}$$

It follows from the construction that the environment cannot introduce two barred letters, otherwise the derivation would never result a terminal word.
Thus, only the sequences consisting of pairs of steps of this type can derive terminal words because otherwise the derivation would never result a terminal word because of the symbol D. (It follows from the construction of Σ').

In the case when λ-rules are allowed we have to modify the above construction in the following way.

- $V_E' = V_E \cup V_E' \cup \{\bar{A} \mid A \to \alpha \in R_{n+1}\} \cup \{D,\ Z\}$, where $D,\ Z \notin V_E$,
- $P_E' = \{ A \to \alpha' \mid A \to \alpha \in P_E \} \cup \{ A \to \bar{A} \mid A \to \alpha \in R_{n+1} \} \cup$
  $\cup \{ A' \to A \mid A \in V_E \} \cup \{ \bar{A} \to D \mid A \to \alpha \in R_{n+1} \} \cup$
  $\cup \{ D \to D \} \cup \{ Z \to \lambda \}$,

- $R_1' = \{A \to \alpha' Z \mid A \to \alpha \in R_1\} \cup \{\bar{A} \to \alpha \mid A \to \alpha \in R_{n+1}\}$,
- $R_i' = \{A \to \alpha' Z \mid A \to \alpha \in R_i\} \cup \{Z \to \lambda\}$ for $2 \leq i \leq n$,
- $\omega' = \omega$,
- $\Delta' = \Delta$.

Here the letters Z guarantee that the agents can act in Σ' in the second simulation step, even if in the first step λ-rules were used. Thus, we have $L(\Sigma, = n+1) = L(\Sigma', = n)$ because of the same reason as in the λ-free case. ■

From the above lemma we obtain the following corollary.

Corollary 3.1.2 *For* $1 \leq n$, $\mathcal{L}(EE^{(\lambda)}(n, = n)) \subseteq \mathcal{L}(EE^{(\lambda)}(1, = 1))$.

There remains only one relation to examine: whether the language classes $\mathcal{L}(EE^{(\lambda)}(n, = n))$ are included in $\mathcal{L}(EE^{(\lambda)}(n+1, = n+1))$ or the inclusion in Lemma 3.1.3 is strict.
The answer depends on the λ-rules. When λ-rules are allowed the two language classes are equal.

Lemma 3.1.4 *For* $1 \leq n$, $\mathcal{L}(EE^{\lambda}(n, = n)) \subseteq \mathcal{L}(EE^{\lambda}(n+1, = n+1))$.

Proof
For any extended simple EG system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega, \Delta)$ we give another extended simple EG system $\Sigma' = (V_E', P_E', R_1', \ldots, R_{n+1}', \omega', \Delta')$ containing λ-rules, such that $L(\Sigma, = n) = L(\Sigma', = n+1)$.
Let Σ' be the following:

- $V_E' = V_E \cup \{D\}$, where $D \notin V_E$,
- $P_E' = P_E \cup \{D \to D\}$,
- $R_i' = R_i$ for $1 \leq i \leq n$,
- $R_{n+1}' = \{D \to D, D \to \lambda\}$,
- $\omega' = \omega D$,
- $\Delta' = \Delta$.

For any $v \in L(\Sigma, = n)$ there is a derivation in Σ of the form

$$\omega \Rightarrow v_1 \Rightarrow \ldots \Rightarrow v_t = v$$

where in the ith step ($1 \leq i \leq t$) the agents R_j use the rules r_{j_i}. Let us consider the following derivation in Σ'

$$\omega D \Rightarrow v_1 D \Rightarrow \ldots \Rightarrow v_{t-1} D \Rightarrow v_t = v$$

where in the ith step ($1 \leq i \leq t-1$) the agents $R_j{}'$ ($1 \leq j \leq n$) apply the rules r_{j_i} corresponding to the above derivation and the agent R'_{n+1} uses the rule $D \to D$; in the last step $R_j{}'$ use the corresponding rules r_{j_t} (for $1 \leq j \leq n$) and R'_{n+1} uses the rule $D \to \lambda$. Thus, we can simulate Σ by Σ'.

Now we show that $L(\Sigma', = n+1) \subseteq L(\Sigma, = n)$. For any $v \in L(\Sigma', = n+1)$ there is a derivation in Σ' of the form

$$\omega' = \omega D \Rightarrow v_1 \Rightarrow \ldots \Rightarrow v_{t-1} \Rightarrow v_t = v$$

where in each derivation step all the $n+1$ agents work. Since v is over $\Delta' = \Delta$, there are no D letters in it. But in the last step $n+1$ agents must work and the last agent can use only the rules $D \to D$ or $D \to \lambda$. Moreover, considering that ω' contains one D and that there is no rule producing new D letters, we get that all words in the above derivation, except v, contain exactly one D letter. We also get that in the first $t-1$ step the $(n+1)$th agent has to use the rule $D \to D$ and it has to use the rule $D \to \lambda$ in the last step.
But in this case we can simulate this derivation of Σ' by the following derivation in Σ:

$$\omega = f(\omega') \Rightarrow f(v_1) \Rightarrow \ldots \Rightarrow f(v_{t-1}) \Rightarrow f(v_t) = v_t = v$$

where f is a homomorphism over $V_E \cup \{D\}$ defined as follows:

$$f(z) = \begin{cases} z & \text{if } z \in V_E \\ \lambda & \text{if } z = D \end{cases}$$

In the above derivation the agents $R_1, R_2, \ldots, R_n$ use the rules corresponding to the derivation of Σ'. ■

We have the following corollary.

Corollary 3.1.3 *For* $1 \leq n$, $\mathcal{L}(EE^{\lambda}(1, = 1)) \subseteq \mathcal{L}(EE^{\lambda}(n, = n))$.

In the λ-free case the inclusion is proper between the language classes $\mathcal{L}(EE(n, = n))$ and $\mathcal{L}(EE(n+1, = n+1))$.

Lemma 3.1.5 *For any* $1 \leq n$ *there exists a language* L, *such that* $L \in \mathcal{L}(EE(n, = n)) \setminus \mathcal{L}(EE(n+1, = n+1))$.

Proof Let L be the finite language $\{ a^n, b^n \}$. It is clear that it can be generated by an extended simple EG system working in mode $= n$. Moreover, it cannot be generated by using mode $= l$ if $l > n$ because either the letters in the axiom are not enough to perform an action of l agents or we should use λ-rules which is not allowed. ■

Concluding the investigation of the derivation mode $= k$, from Corollary 3.1.1, Corollary 3.1.2 and Corollary 3.1.3 we obtain the following theorem. It shows that if λ-rules are allowed, neither the number of the agents being in the system nor the number of the agents working in a step have any influence on the generated language class.

Theorem 3.1 *For* $1 \leq k \leq n$, $\mathcal{L}(EE^{\lambda}(n, = k)) = \mathcal{L}(EE^{\lambda}(1, = 1))$.

In the λ-free case we use the notation $\mathcal{L}(EE(= k))$ for the class of the languages which can be generated by extended simple EG systems working in mode $= k$, without λ-rules. From Corollary 3.1.1 we have $\mathcal{L}(EE(n, = k)) = \mathcal{L}(EE(= k))$ for $1 \leq k \leq n$.
Using Corollary 3.1.1, Lemma 3.1.3 and Lemma 3.1.5, we can summarize the results of the λ-free case in the following theorem.

Theorem 3.2
$\mathcal{L}(EE(= 1)) = \mathcal{L}(EE(n_1, = 1)) \supset \mathcal{L}(EE(= 2)) = \mathcal{L}(EE(n_2, = 2)) \supset \ldots \supset \mathcal{L}(EE(= k)) = \mathcal{L}(EE(n_k, = k)) \supset \ldots$
where $n_i \geq i$ *for* $1 \leq i$.

3.2 The hierarchy of language classes $\mathcal{L}(EE^{(\lambda)}(n, \leq k))$

In this section we examine the hierarchy of the language classes generated by using derivation mode $\leq k$. In most of the cases the statements and proofs of the case $= k$ are valid for the derivation mode $\leq k$ as well, in such cases we refer to the previous section.

The first lemma can be proved by the same construction as in Lemma 3.1.1.

Lemma 3.2.1 *For* $1 \leq k \leq n \leq m$, $\mathcal{L}(EE^{(\lambda)}(n, \leq k)) \subseteq \mathcal{L}(EE^{(\lambda)}(m, \leq k))$.

The proof of the second lemma uses the same construction and follows the same idea as the proof of Lemma 3.1.2, thus we do not give the detailed proof here.

Lemma 3.2.2 *For* $1 \leq k \leq n$, $\mathcal{L}(EE^{(\lambda)}(n+1, \leq k)) \subseteq \mathcal{L}(EE^{(\lambda)}(n, \leq k))$.

From these two above lemmas we obtain the following corollary:

Corollary 3.2.1
For $1 \leq k \leq n$, $\mathcal{L}(EE^{(\lambda)}(n, \leq k)) = \mathcal{L}(EE^{(\lambda)}(k, \leq k))$.

Thus, it is enough, similarly to the case $= k$, to examine the relation of the language classes $\mathcal{L}(EE^{(\lambda)}(k, \leq k))$

We have the same result as in the case of the derivation mode $= k$. For proving this statement, we can use the same construction and explanation as in Lemma 3.1.3.

Lemma 3.2.3 *For* $1 \leq n$, $\mathcal{L}(EE^{(\lambda)}(n+1, \leq n+1)) \subseteq \mathcal{L}(EE^{(\lambda)}(n, \leq n))$.

The above lemmas give the following corollary.

Corollary 3.2.2 *For* $1 \leq n$, $\mathcal{L}(EE^{(\lambda)}(n, \leq n)) \subseteq \mathcal{L}(EE^{(\lambda)}(1, \leq 1))$.

In the case of the derivation mode $= k$ the language classes formed a collapsing hierarchy if and only if λ-rules were allowed in the system. If we consider the derivation mode $\leq k$, we have different results.

Lemma 3.2.4 *For* $1 \leq k \leq n$, $\mathcal{L}(EE^{(\lambda)}(n, \leq n)) \subseteq \mathcal{L}(EE^{(\lambda)}(n+1, \leq n+1))$.

Proof
For any extended simple EG system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega, \Delta)$ we give another extended simple EG system $\Sigma' = (V_E', P_E', R_1', \ldots, R_{n+1}', \omega', \Delta')$, such that $L(\Sigma, \leq n) = L(\Sigma', \leq n+1)$. Let Σ' be the following:

- $V_E' = V_E \cup \{D\}$, where $D \notin V_E$,
- $P_E' = P_E \cup \{D \to D\}$,
- $R_i' = R_i$ for $1 \leq i \leq n$,
- $R_{n+1}' = \{D \to D\}$,
- $\omega' = \omega$,
- $\Delta' = \Delta$.

The generated language of Σ in mode $\leq n$ is the same as that of Σ' in mode $\leq n+1$ because in Σ' an $\leq n+1$ derivation means an $\leq n$ derivation, considering the structure of Σ'. ■

From the above lemma we have the following corollary.

Corollary 3.2.3 *For* $1 \leq n$, $\mathcal{L}(EE^{(\lambda)}(1, \leq 1)) \subseteq \mathcal{L}(EE^{(\lambda)}(n, \leq n))$

Concluding the investigation of the case $\leq k$, from Corollary 3.2.1, Corollary 3.2.2 and Corollary 3.2.3 we obtain that the hierarchy of the generated language classes is a collapsing one, both with and without λ-rules.

Theorem 3.3 *For* $1 \leq k \leq n$,

$$\mathcal{L}(EE^{(\lambda)}(n, \leq k)) = \mathcal{L}(EE^{(\lambda)}(1, \leq 1))$$

We have already seen that only the language classes $\mathcal{L}(EE^{\lambda}(1, = 1))$, $\mathcal{L}(EE(k, = k))$, $\mathcal{L}(EE^{\lambda}(1, \leq 1))$ and $\mathcal{L}(EE(1, \leq 1))$ are interesting, the other classes are equal to one of these language families.
Thus, the only remaining thing is to examine the relation of these special language classes.
The following lemma holds by the definition of the derivation modes ≤ 1 and $= 1$.

Lemma 3.2.5 $\mathcal{L}(EE^{(\lambda)}(1, = 1)) = \mathcal{L}(EE^{(\lambda)}(1, \leq 1))$

Moreover, language classes generated without λ-rules are included in the language classes generated with the same parameters but with λ-rules.

Lemma 3.2.6 $\mathcal{L}(EE^{\lambda}(1, = 1)) \supset \mathcal{L}(EE(1, = 1))$

Proof The inclusion holds by definition; it is proper because languages containing at least one non-empty word and the empty word λ cannot be generated without λ-rules. ■

4 Summary and open problems

We can summarize the results of the paper in the form of a diagram. In this diagram, a straight arrow indicates a proper inclusion; the class the arrow leaves is included in the class the arrow points at.
We use the notation n_i for $1 \leq i$, where n_i denotes an arbitrary positive integer, such that $i \leq n_i$.

$$\begin{array}{ccccccc}
\mathcal{L}(EE^{\lambda}(n, \leq k)) & = & \mathcal{L}(EE^{\lambda}(1, \leq 1)) & = & \mathcal{L}(EE^{\lambda}(1, = 1)) & = & \mathcal{L}(EE^{\lambda}(n, = k)) \\
 & & & & \uparrow & & \\
\mathcal{L}(EE(n, \leq k)) & = & \mathcal{L}(EE(1, \leq 1)) & = & \mathcal{L}(EE(1, = 1)) & = & \mathcal{L}(EE(n_1, = 1)) \\
 & & & & \uparrow & & \\
 & & & & \mathcal{L}(EE(2, = 2)) & = & \mathcal{L}(EE(n_2, = 2)) \\
 & & & & \uparrow & & \\
 & & & & \vdots & & \\
 & & & & \uparrow & & \\
 & & & & \mathcal{L}(EE(k, = k)) & = & \mathcal{L}(EE(n_k, = k))
\end{array}$$

The main result of this contribution is that while in the non-extended case the size parameters (n and k) and the mode of the derivation have influence on the generated language classes (see [4]), in the extended case the hierarchy is a collapsing one.

There have remained some open problems. In this paper we investigated only the relation of the language classes generated by extended simple EG systems with different parameters and derivation modes. The precise place of these language classes in a traditional hierarchy, that is their relation to the Lindenmayer or the Chomsky language classes will be the subject of future investigation. Some results concerning this question were given in [5], it was shown there that the λ-free classes are between the language families $E0L$ and MAT_{ac}.

More new questions arise if we consider another definition (different from Definition 2.3) for the derivation mode $\leq k$. By Definition 2.3, in every derivation steps at least one agent has to work. By omitting this condition and allowing that only the environment works in a derivation step, we obtain a new definition for this derivation mode and we obtain new language classes as well.

Moreover, similarly to the $\leq k$ mode we can give the definition of the $\geq k$ derivation.
The relation of the language classes defined by these derivation modes and the language classes generated by the original ones can be examined in the future.

References

[1] J. Csima. Formal language theoretic models of ecosystems (In Hungarian). Master's thesis, ELTE, Budapest, 1997.

[2] E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová, and Gh. Păun. Eco(-grammar) systems. A preview. In R. Trappl, editor, *Proc 12th European Meeting on Cybernetics and System Research*, pages 941–948. World Sci. Publ., Singapore, 1994.

[3] E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová, and Gh. Păun. A grammatical approach to likelife interactions. *Artificial Life*, 3:1–28, 1997.

[4] E. Csuhaj-Varjú and A. Kelemenová. Team behaviour in simple eco-grammar systems. *Theoretical Computer Science*, 209:213–224, 1998.

[5] J. Dassow and V. Mihalache. Eco-grammar systems, matrix grammars and E0L systems. In Gh. Păun, editor, *Artificial Life: grammatical models*, pages 210–226. Black Sea Univ. Press, Bucharest, 1995.

[6] Gh. Păun, editor. *Artificial Life: grammatical models.* Black Sea Univ. Press, Bucharest, 1995.

[7] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L-systems.* Academic Press, 1980.

[8] A. Salomaa. *Formal languages.* Academic Press, 1973.

[9] M. H. ter Beek. Teams in grammar systems. Master's thesis, Leiden University, Leiden, 1996.

[10] M. H. ter Beek. Teams in grammar systems: Hybridity and weak rewriting. *Acta Cybernetica*, 12(4):427–444, 1996.

Acta Cybernetica **13** (1998) 375-384.

Descriptional Complexity of Multi-Continuous Grammars

Alexander Meduna *

Abstract

The present paper discusses multi-continuous grammars and their descriptional complexity with respect to the number of nonterminals. It proves that six-nonterminal multi-continuous grammars characterize the family of recursively enumerable languages. In addition, this paper formulates an open problem area closely related to this characterization.

Key Words: multi-continuous grammars; descriptional complexity; nonterminals; recursively enumerable languages.

1 Introduction

The language theory has intensively and systematically investigated the descriptional complexity of grammars (see Chapter 4 in [1] and references therein). This investigation has achieved several characterizations of the family of recursively enumerable languages by various grammars with a reduced number of nonterminals (see [4] through [6]).

The present paper discusses the descriptional complexity of multi-continuous grammars (see [3]). It proves that six-nonterminal multi-continuous grammars characterize the family of recursively enumerable languages. In its conclusion, this paper points out some open problems closely related to this characterization.

2 Definitions

This paper assumes that the reader is familiar with the formal language theory, including selective substitution grammars (see Chapter 10 in [1])).

Let Σ be an alphabet. The cardinality of Σ is denoted by $Card(\Sigma)$. Σ^* represents the free monoid generated by si under the operation of concatenation. The unit of Σ^* is denoted by ε. Set $\Sigma^+ = \Sigma^* - \Sigma^* - \{\varepsilon\}$; algebraically, Σ^+ is the free semigroup generated by Σ under the operation of concatenation. For $w \in \Sigma^*$, $|w|$ denotes the length of w and $subword(w)$ is defined as $subword(w) = \{x : x \in V^*$ and x is a subword of $w\}$.

*Computing Center at Technical University of Brno, Udolni 19, Brno 60200, Czech Republic

The **bold** symbols have special meaning hereafter. If a is a symbol, then **a** means that the original symbol, a, is *activated.* Analogously, for an alphabet Σ,

$$\mathbf{\Sigma} = \{\mathbf{a} : a \in \Sigma\} \text{ and } \{\mathbf{x} : x \in \Sigma^+\}.$$

Define the homomorphism, ι, from $(\mathbf{\Sigma} \cup \Sigma)^*$ to Σ^* as

$$\iota(\mathbf{a}) = a \text{ and } \iota(a) = a$$

for all $a \in \Sigma$.

An *EOS system* is quadruple

$$E = (\Sigma, P, S, T),$$

where Σ is an alphabet, $T \subseteq \Sigma, S \in \Sigma - T$, and P is a finite substitution on $\Sigma + *$. An *EOS-based s-grammar,* G, is a quintuple

$$G = (\Sigma, P, S, T, K),$$

where Σ, P, S, and T have the same meaning as in an EOS system, and $K \subseteq (\mathbf{\Sigma} \cup \Sigma)^*$. Let $u, v \in \Sigma^*$. G *directly derives* v *from* u, symbolically denoted as

$$u \Rightarrow v,$$

if either $u = S$ and $v \in P(S)$ or there exists a natural number, n, so

1. $u = a_1 \dots a_n$ with $a_i \in T$ for all $i = 1, \dots, n$
2. $w = b_1 \dots b_n, w \in K$, and $\iota(w) = u$
3. $v = x_1 \dots x_n$ with $x_i \in P(a_i)$ if $b_i \in \mathbf{\Sigma}$, and $x_i = a_i$ if $b_i \in \Sigma$ for each $i = 1, \dots, n$.

Instead of $x \in P(a)$, this paper writes $a \to x$ hereafter. In the standard manner, extend $\Rightarrow$to $\Rightarrow^n$, where $n \geq 0$. Based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The *language of* $G, L(G)$,

is defined as

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}.$$

Let m be a natural number, and let $G = (\Sigma, P, S, T, K)$ be an EOS-based s-grammar. G is an *m-continuous grammar* if for some $n \geq 1$,

$$K = K_1 \cup \dots \cup K_n$$

so that for $i = 1, \dots, n$,

$$K_i = \Omega_1 \mathbf{\Pi_1} \Omega_2 \dots \Omega_m \mathbf{\Pi_m} \Omega_{m+1},$$

where

$$\Omega_j \in \{V^* : V \subseteq \Sigma\} \text{ for } j = 1, \dots, m+1$$

$\Pi_k \in \{W^+ : W \subseteq \Sigma\}$ for $k = 1, \ldots, m$.

G is a *multi-continuous grammar* if G represents an m-continuous *grammar* for some $m \geq 1$. A *queue grammar* (see [2]) is a sixtuple, $Q = (V, T, W, F, R, g)$, where V and W are alphabets satisfying $V \cap W = \emptyset, T \subseteq V, F \subseteq V, F \subseteq W, R \in (V-T)(W-F)$, and $g \subseteq (V \times (W-F)) \times (V^* \times W)$ is a finite relation such that for any $a \in V$, there exists an element $(a, b, x, c) \in g$. If there exist $u, v \in V^*W, a \in V, r, z \in V^*$, and $b, c \in W$ such that $(a, b, z, c) \in g, u = arb$, and $v = rzc$, then Q directly derives v from u, denoted by $u \Rightarrow v$. In the standard manner, define $\Rightarrow^n, \Rightarrow^+$, and $\Rightarrow^*$. A derivation of the form $R \Rightarrow^* wf$ with $w \in T^*$ and $f \in F$ is a successful derivation. The language of $QL(Q)$, is defined as $L(Q) = \{w \in T^* : R \Rightarrow^* wf \text{ where } f \in F\}$.

3 Results

The present section demonstrates that the family of recursively enumerable languages equals the family of languages g 1 by six-nonterminal multicontinuous grammars.

Lemma 1 *Let*

$$Q = (V, T, W, FR, g)$$

be a queue grammar. Then, there exists a six-nonterminal multi-continuous grammar, G, satisfying

$$L(G) - \{\varepsilon\} = L(Q) - \{\varepsilon\}.$$

Proof: Let

$$Q = (V, T, W, F, R, g)$$

be a queue grammar. Without any loss of generality, assume that

$$(V \cup W) \cap \{0, 1, 2, 3, X, Y\} = \emptyset.$$

Construction:

For some $n \geq 2^{\#(V \cup W)}$, introduce the following four mappings $-\beta, \rho, \sigma$, and δ:

1. Define an injection β from $(V \cup W)$ to $(\{0,1\}\{3\})^n$. In the standard manner, extend β so it is defined from $(V \cup W)^*$ to $((\{0,1\}\{3\})^n)^*$. β^{-1} represents the inverse of β.

2. Let ρ be the mapping from $(\{0,1\}\{3\})^n((\{0,1\}\{3\})^n \cup T)^*$ to $((\{0,1\}\{3\})^n \cup T)^*(\{0,1\}\{3\})^n$ defined as

   $$\rho(ax) = xa$$

   for all $a \in (\{0,1\}\{3\})^n$ and $x \in ((\{0,1\}\{3\})^n \cup T)^*$.

3. Let σ be the mapping from $(T \cup \{0,1,2,3\})^*$ to $(T \cup \{0,1,3\})^*$ defined as

   $$\sigma(a) = a \text{ for all } a \in T \cup \{0,1,3\} \text{ and } \sigma(2) = \varepsilon.$$

4. Let δ be the mapping from $\{0,1,3\}^*$ to $\{X,Y,3\}^*$ defined as

$$\delta(0) = X, \delta(1) = X \text{ and } \delta(3) = 3.$$

Set

$$m = \max\{|\beta(x)| : (a,b,x,c) \in g \text{ and some } a \in W - F, c \in W, \text{ and } b \in V\} + 6n + 2.$$

Define the following m-continuous grammar

$$G = (T \cup \{0,1,2,3,X,Y\}, P, 2, T, K),$$

where

$$\begin{aligned}
P = {} & \{2 \to \beta(b)2\beta(a)X^{m-2|\beta(b)\beta(a)|-2}2 : a \in V - T, b \in W - F, ab = R\} \\
\cup {} & \{a \to a : a \in T \cup \{0,1,2,3\}\} \\
\cup {} & \{3 \to 32, 2 \to \varepsilon\} \\
\cup {} & \{i \to \delta(i) : i = 0,1,3\} \\
\cup {} & \{a \to \varepsilon : a \in \{X,Y,3\}\} \\
\cup {} & \{2 \to X^j 2 : j = 1,\ldots,m-4n-2\} \\
\cup {} & \{2 \to X^j : j = 1,\ldots,m-2n-1\} \\
\cup {} & \{2 \to \beta(c)2 : c \in W\} \\
\cup {} & \{2 \to \beta(x)X^{m-|\beta(abcx)|-2}2 : x \in V^*, \text{ and } (a,b,x,c) \in g, \text{ where} \\
& a, c \in W - F \text{ and } b \in V\} \\
\cup {} & \{2 \to \beta(x)X^{m-|\beta(abcx)y|-2}2 : x \in V^*, y \in T^+, \text{ and } (a,b,xy,c) \in g, \text{ for some} \\
& a \in W - F, c \in W, \text{ and } b \in V\} \\
\cup {} & \{2 \to yX^{m-|\beta(abc)y|-2}2 : y \in T^*, \text{ and } (a,b,y,c) \in g, \text{ for some} \\
& a \in W - F, c \in W, \text{ and } b \in V\}.
\end{aligned}$$

Furthermore,

$$K = K_1 \cup K_2 \cup K_3 \cup K_4 \cup K_5 \cup K_6$$

where K_1 through K_6 are constructed as follows. Initially, set

$$K_i = \emptyset$$

for $i = 1,\ldots,6$. Then, extend K_1 through K_6 in the following way.

A. If

$$(a,b,x,c) \in g, \text{ where } b,c \in W, a \in V, \text{ and } x \in V^*$$

then

$$\begin{aligned}
K_1 := K_1 \cup \{ & \{\mathbf{b_1}\}^+\{3\}^+ \ldots \{\mathbf{b_n}\}^+\{3\}^+\{2\}^+\{\mathbf{a_1}\}^+\{3\}^+ \ldots \{\mathbf{a_n}\}^+\{3\}^+ \\
& (\{0,1,3\} \cup T)^*\mathbf{H_1} \ldots \mathbf{H_{m-|\beta(ba)|-2}}\{2\}^+\},
\end{aligned}$$

where
$a_i, b_i \in \{0,1\}$ for $i = 1, \ldots, n$
$a_1 3 \ldots a_n 3 = \beta(a)$
$b_1 3 \ldots b_n 3 = \beta(b)$
$H_j = \{X\}^+$, for all $j = 1, \ldots, m - 4n - 2$

$$K_2 := K_2 \cup \{\{\mathbf{b_1}\}^+\{\mathbf{3}\}^+ \ldots \{\mathbf{b_n}\}^+\{\mathbf{3}\}^+\{\mathbf{a_1}\}^+\{\mathbf{3}\}^+ \ldots \{\mathbf{a_n}\}^+\{\mathbf{3}\}^+\{\mathbf{2}\}^+ (\{0,1,3\} \cup T)^*\mathbf{H_1} \ldots \mathbf{H_{m-|\beta(ba)|-2}}\{\mathbf{2}\}^+\},$$

where
$a_i, b_i \in \{0,1\}$ for $i = 1, \ldots, n$
$a_1 3 \ldots a_n 3 = \beta(a)$
$b_1 3 \ldots b_n 3 = \beta(b)$
$H_j = \{X\}^+$, for all $j = 1, \ldots, m - 4n - 2$

$$K_3 := K_3 \cup \{\delta\{(\mathbf{b_1})\}^+\{\mathbf{3}\}^+ \ldots \{\delta(\mathbf{b_n})\}^+\{\mathbf{3}\}^+\{\delta(\mathbf{a_1})\}^+\{\mathbf{3}\}^+ \ldots \{\delta(\mathbf{a_n})\}^+\{\mathbf{3}\}^+\{\mathbf{c_1}\}^+\{\mathbf{3}\}^+ \ldots \{\mathbf{c_n}\}^+\{\mathbf{3}\}^+\{\mathbf{2}\}^+(\{0,1,3\}^*\{\mathbf{d_1}\}^+\{\mathbf{3}\}^+ \ldots \{\mathbf{d_{|x|}}\}^+\{\mathbf{3}\}^+\mathbf{H_1} \ldots \mathbf{H_{m-|\beta(bacx)|-2}}\{\mathbf{2}\}^+\},$$

where
$a_i, b_i, c_i, d_i \in \{0,1\}$, for $i = 1, \ldots, n$
$a_1 3 \ldots a_n 3 = \beta(a)$
$b_1 3 \ldots b_n 3 = \beta(b)$
$c_1 3 \ldots c_n 3 = \beta(c)$ for some $c \in V$
$d_1 3 \ldots d_{|x|} 3 = \beta(x)$
$H_j = \{X\}^+$, for all $j = 1, \ldots, m - |\beta(bacx)| - 2$.

B. If

$$x \in V^*, y \in T^+, \text{ and } (a, b, xy, c) \in g \text{ for some } b, c \in W \text{ and } a \in V$$

then

$$K_4 := K_4 \cup \{\{\delta(\mathbf{b_1})\}^+\{\mathbf{3}\}^+ \ldots \{\delta(\mathbf{b_n})\}^+\{\mathbf{3}\}^+\{\delta(\mathbf{a_1})\}^+\{\mathbf{3}\}^+ \ldots \{\delta(\mathbf{a_n})\}^+\{\mathbf{3}\}^+\{\mathbf{c_1}\}^+\{\mathbf{3}\} \ldots \{\mathbf{c_n}\}^+\{\mathbf{3}\}^+\{\mathbf{2}\}^+\{0,1,3\}^*\{\mathbf{d_1}\}^+\{\mathbf{3}\}^+ \ldots \{\mathbf{d_{|x|}}\}^+\{\mathbf{3}\}^+\{\mathbf{e_1}\}^+ \ldots \{\mathbf{e_{|y|}}\}^+\mathbf{H_1} \ldots \ldots \mathbf{H_{m-|\beta(bacx)y|-2}}\{\mathbf{2}\}^+\},$$

where
$a_i, b_i \in \{0,1\}$, for $i = 1, \ldots, n$
$a_1 3 \ldots a_n 3 = \beta(a)$
$b_1 3 \ldots b_n 3 = \beta(b)$
$c_1 3 \ldots c_n 3 = \beta(c)$ for some $c \in V$
$d_1 3 \ldots d_{|x|} 3 = \beta(x)$
$e_1 \ldots e_{|y|} = y$
$H_j = \{X\}^+$, for all $j = 1, \ldots, m - |\beta(x)| - |y| - 6n - 2$.

C. If

$$x \in T^* \text{ and } (a, b, x, c) \in g \text{ for some } b, c \in W \text{ and } a \in V$$

then

$$\begin{aligned} K_5 := K_5 \cup \{&\{\delta(\mathbf{b_1})\}^+\{\mathbf{3}\}^+ \dots \{\delta(\mathbf{b_n})\}^+\{\mathbf{3}\}^+\{\delta(\mathbf{a_1})\}^+\{\mathbf{3}\}^+ \dots \\ &\{\delta(\mathbf{a_n})\}^+\{\mathbf{3}\}^+\{\mathbf{c_1}\}^+\{\mathbf{3}\} \dots \{\mathbf{c_n}\}^+\{\mathbf{3}\}^+\{\mathbf{2}\}^+\{0,1,3\}^* \\ &\mathbf{T}^+\{\mathbf{e_1}\}^+ \dots \{\mathbf{e}^+_{|\mathbf{x}|}\mathbf{T}^*\mathbf{H_1} \dots \mathbf{H_{m-|\beta(bac)x|-6n-3}}\{\mathbf{2}\}^+\}, \end{aligned}$$

where
$a_i, b_i \in \{0, 1\}$, for $i = 1, \dots, n$
$a_1 3 \dots a_n 3 = \beta(a)$
$b_1 3 \dots b_n 3 = \beta(b)$
$c_1 3 \dots c_n 3 = \beta(c)$ for some $c \in V$
$e_1 \dots e_{|x|} = x$
$H_j = \{X\}^+$, for all $j = 1, \dots, m - |x| - 6n - 3$

D. If

$$b \in F$$

then

$$K_6 := K_6 \cup \{\{\delta(\mathbf{b_1})\}^+\{\mathbf{3}\}^+ \dots \{\delta(\mathbf{b_n})\}^+\{\mathbf{3}\}^+\mathbf{H_1} \dots \mathbf{H_{m-2n-1}}\mathbf{T}^+T^*\},$$

where
$b_i \in \{0, 1\}$, for all $i = 1, \dots, n$
$b_1 3 \dots b_n 3 = \beta(b)$
$H_j = \{X\}^+$, for all $j = 1, \dots, m - |\beta(b)| - 1$.

Main Idea:

Observe that G derives no sentential form that contains a subword consisting of two identical nonterminals. Considering this essential property, examine the construction of G to see that every successful derivation simulates a successful derivation in Q. To give an insight into this simulation in greater detail, assume that Q makes this derivation step

$$avb \Rightarrow vxc$$

according to $(a, b, x, c) \in g$. By using selectors constructed in A, G simulates $avb \Rightarrow vxc$ by making the following three steps.

$$\begin{aligned} \beta(b)2\beta(av)X^{m-|\beta(ba)|-2}2 &\Rightarrow \beta(ba)2\beta(ba)2\beta(v)X^{m-|\beta(ba)|-2}2 \\ &\Rightarrow \delta(\beta(ba))\beta(c)2\beta(vx)X^{m-|\beta(bacx)|-2}2 \\ &\Rightarrow \beta(c)2\beta(vx)X^{m-4n-2}2. \end{aligned}$$

By analogy with these steps, G uses selectors constructed in B and C to simulate $Q's$ derivation steps that produce terminals appearing in the generated word. Finally, it uses a selector constructed in D to complete the simulation. As a result, $L(Q) = L(G)$.

Formal Proof (Sketch):
Hereafter, by

$$u \Rightarrow v \ [i]$$

in G, where $i \in \{1, \ldots, 6\}$, this proof symbolically expresses that G makes $u \Rightarrow v$ by using a component from K_i. For brevity, the rest of this proof omits some details, which the reader can easily fill in. Examine K to see that in G, every successful derivation, $2 \Rightarrow^+ v$ with $v \in T^+$, has this form

$$\begin{array}{lllllllll}
2 & \Rightarrow & x_0 & & & & & & \\
 & \Rightarrow & x_{1_1} & [1] & \Rightarrow & x_{1_2} & [2] & \Rightarrow & x_{1_3} \ [3] \\
 & \Rightarrow & x_{2_1} & [1] & \ldots & & & & \\
 & \ldots & & & & & & & \\
 & \Rightarrow & x_{t_1} & [1] & \Rightarrow & x_{t_2} & [2] & \Rightarrow & x_{t_3} \ [3] \\
 & \Rightarrow & y_1 & [1] & \Rightarrow & y_2 & [2] & \Rightarrow & y_3 \ [4] \\
 & \Rightarrow & z_{1_1} & [1] & \Rightarrow & z_{1_2} & [2] & \Rightarrow & z_{1_3} \ [5] \\
 & \Rightarrow & z_{2_1} & [1] & \ldots & & & & \\
 & \ldots & & & & & & & \\
 & \Rightarrow & z_{h_1} & [1] & \Rightarrow & z_{h_2} & [2] & \Rightarrow & z_{h_3} \ [5] \\
 & \Rightarrow & r & [1] & \Rightarrow & v & [6], & &
\end{array}$$

where

(i) $x_0 = \beta(b)2\beta(a)X^{m|\beta(ba)|-2}2$ with $ab = R$

(ii) t is a non-negative integer, and for all $i = 0, \ldots, t$, there exist $(a, b, v, c) \in g$ and $u \in V^*$ so that

$$\begin{aligned}
x_{i_1} &= \beta(ba)2\beta(u)X^{m-|\beta(ba)|-2}2 \\
x_{i_2} &= \delta(\beta(ba))\beta(c)2\beta(uv)X^{m-|\beta(bacv)|-2}2 \\
x_{i_3} &= \beta(c)2\beta(uv)X^{m-2|\beta(c)|-2}2
\end{aligned}$$

(iii) there exist $w \in V^*$ and $(a, b, vu, c) \in g$ where $v \in V^*$ and $u \in T^+$, so that

$$\begin{aligned}
y_1 &= \beta(ba)2\beta(w)X^{m-|\beta(ba)|-2}2 \\
y_2 &= \delta(\beta(ba))\beta(c)2\beta(wv)uX^{m-|\beta(bacv)u|-2}2 \\
y_3 &= \beta(c)2\beta(wv)uX^{m-2|\beta(c)|-2}2
\end{aligned}$$

(iv) h is a non-negative integer, and for all $i = 0, \ldots, h$, there exist $u \in V^*, w \in T^+$, and $(a, b, v, c) \in g$ with $v \in T^*$ so that

$$\begin{aligned}
z_{i_1} &= \beta(ba)2\beta(u)wX^{m-|\beta(ba)|-2}2 \\
z_{i_2} &= \delta(\beta(ba))\beta(c)2\beta(u)wvX^{m-|\beta(bac)v|-2}2 \\
z_{i_3} &= \beta(c)2\beta(u)wvX^{m-2|\beta(c)|-2}2
\end{aligned}$$

(v) $r = \delta(\beta(b))vX^{m-|\beta(c)|-1}$ with $b \in F$.

Observe that there also exists the following derivation

$$\begin{aligned} R &\Rightarrow \rho(\beta^{-1}(\sigma(x_{1_3})))\ldots\Rightarrow \rho(\beta^{-1}(\sigma(x_{h_3}))) \\ &\Rightarrow \rho(\beta^{-1}(\sigma(y_3))) \\ &\Rightarrow \rho(\beta^{-1}(\sigma(x_{1_3})))\ldots\Rightarrow \rho(\beta^{-1}(\sigma(x_{h_3}))) \\ &\Rightarrow \rho(\beta^{-1}(\sigma(r))) \end{aligned}$$

in Q. Notice that $\rho(\beta^{-1}(\sigma(r))) = v$. Thus, if in $G, 2 \Rightarrow^* v$ with $v \in T^+$, then $v \in L(Q)$; therefore,

$$L(G) - \{\varepsilon\} \subseteq L(Q) - \{\varepsilon\}.$$

Notice that in Q, every successful derivation, $R \Rightarrow^* vf$ with $v \in T^+$ and $f \in F$, has this form

$$\begin{aligned} R &\Rightarrow^* d_1 d_2 \ldots d_n y_1 c_1 \\ &\Rightarrow d_2 \ldots d_n y_1 y_2 c_2 \\ &\ldots \\ &\Rightarrow d_n y_1 y_2 \ldots y_n c_n \\ &\Rightarrow y_1 y_2 \ldots y_n f, \end{aligned}$$

where

n is a natural number
$d_k \in V$, for $k = 1, \ldots, n$
$v = y_1 y_2 \ldots y_n$
$y_1 \neq \varepsilon$
$y_i \in T^*$, for $i = 2, \ldots, n$
$c_j \in W - F$, for $j = 1, \ldots, n$
$f \in F$.

Consider any derivation expressed in this way in Q, and demonstrate that there also exists

$$2 \Rightarrow^+ v$$

in G (a detailed version of this demonstration is left to the reader). Thus

$$L(Q) - \{\varepsilon\} \subseteq L(G) - \{\varepsilon\}.$$

As $L(G) - \{\varepsilon\} \subseteq L(Q) - \{\varepsilon\}$ and $L(Q) - \{\varepsilon\} \subseteq L(G) - \{\varepsilon\}$,

$$L(Q) - \{\varepsilon\} = L(G) - \{\varepsilon\}.$$

Because G has only the six nonterminals $0, 1, 2, 3, X$, and Y, Lemma 1 holds. □

Theorem 1 *The family of languages generated by six-nonterminal multi-continuous grammars coincides with the family of recursively enumerable languages.*

Proof: Obviously, every language generated by a six-nonterminal multi-continuous grammar represents a recursively enumerable language. The rest of this proof demonstrates that every recursively enumerable language is generated by a six-non terminal multi-continuous grammar.

Let L be a recursively enumerable language. Then, there exists a queue grammar, Q, such that $L(Q) = L$ (see Theorem 2.1 in [2]). By Lemma 1, there exists a six-nonterminal multi-continuous grammar,

$$G = (T \cup \{0, 1, 2, 3, X, Y\}, P, 2, T, K),$$

satisfying $L(Q) - \{\varepsilon\} = L(G) - \{\varepsilon\}$. Consider the six-nonterminal multi-continuous grammar, G', defined as

$$G' = (T \cup \{0, 1, 2, 3, X, Y\}, P \cup P', 2, T, K)$$

with

$$P' = \{2 \to \varepsilon\} \text{ if } \varepsilon \in L(Q), \text{ and } P' = \emptyset \text{ if } \varepsilon \notin L(Q).$$

Observe that $L(G) - \{\varepsilon\} = L(G') - \{\varepsilon\}$. Because $L(Q) - \{\varepsilon\} = L(G) - \{\varepsilon\}, L(Q) - \{\varepsilon\} = L(G') - \{\varepsilon\}$. Furthermore, by the definition of $P', \varepsilon \in L(Q)$ if and only if $\varepsilon \in L(G')$. Therefore,

$$L(G') = L(Q).$$

As $L(Q) = L$,

$$L = L(G').$$

Therefore, this theorem holds. □

Consider i-nonterminal multi-continuous grammars, where $i = 1, \ldots, 5$. What is their generative power?

Acknowledgement: The author is indebted to the anonymous referee for useful comments concerning the first version of this paper.

References

[1] Dassow, J. and Paun, G.: *Regulated Rewriting in Formal Language Theory.* Springer, New York,1989.

[2] Kleijn, H. C. M. and Rozenberg, G.: "On the Generative Power of Regular Pattern Grammars," *Acta Informatica,* Vol. 20, pp. 391-411, 1983.

[3] Kleijn, H. C. M. and Rozenberg, G.: "Multi Grammars," *International Journal of Computer Mathematics,* Vol.12, pp. 177-201, 1983.

[4] Meduna, A. : "Six-Nonterminal Multi-Sequential Grammars Characterize the Family of Recursively Enumerable Languages," *International Journal of Computer Mathematics,* Vol. 65, pp. 179-189,1997.

[5] Meduna, A. : On the Number of Nonterminals in Matrix Grammars with Leftmost Derivations, in Paun, G. and Salomaa, A. (ed.), *New Trends in Formal Languages,* Lecture Notes of Computer Science 1218, 1997, 27 - 38

[6] Paun, Gh. : "Six Nonterminals are Enough for Generating each R. E. Language by a Matrix Grammar," *International Journal of Computer Mathematics,* Vol. 15, pp. 23-37, 1984.

Received May, 1997

Acta Cybernetica **13** (1998) 385-403.

Decompositions of automata and transition semigroups *

Tatjana Petković† Miroslav Ćirić‡ Stojan Bogdanović§

Abstract

The purpose of this paper is to describe structural properties of automata whose transition semigroups have a zero, left zero, right zero or bi-zero, or are nilpotent extensions of rectangular bands, left zero bands or right zero bands, or are nilpotent. To describe the structure of these automata we use various well-known decomposition methods of automata theory – direct sum decompositions, subdirect and parallel decompositions, and extensions of automata. Automata that appear as the components in these decompositions belong to some well-known classes of automata, such as directable, definite, reverse definite, generalized definite and nilpotent automata. But, we also introduce some new classes of automata: generalized directable, trapped, one-trapped, locally directable, locally one-trapped, locally nilpotent and locally definite automata. We explain relationships between the classes of all these automata.

Keywords: automaton, transition semigroup, direct sum decomposition, directable automata, trapped automata, generalized directable automata, locally directable automata, generalized varieties.

1. Introduction and preliminaries

Transition semigroups of automata were first defined and studied by V. M. Glushkov in [16], 1961. The systematic study of relationships between the structure of automata and their transition semigroups was initiated by I. Peák in [23], 1964, and [24], 1965, and after that many authors worked on this important topic. Many of the results concerning this topic were collected in the book of F. Gécseg and I. Peák [14], in 1972, and in some other books.

*Supported by Grant 04M03B of RFNS through Math. Inst. SANU.

†University of Niš, Faculty of Philosophy, Ćirila i Metodija 2, P. O. Box 91, 18000 Niš, Yugoslavia, e-mail: tanjapet@archimed.filfak.ni.ac.yu

‡University of Niš, Faculty of Philosophy, Ćirila i Metodija 2, P. O. Box 91, 18000 Niš, Yugoslavia, e-mail: mciric@archimed.filfak.ni.ac.yu

§University of Niš, Faculty of Economics, Trg VJ 11, P. O. Box 121, 18000 Niš, Yugoslavia, e-mail: sbogdan@archimed.filfak.ni.ac.yu

The main aim of the present paper is to investigate structural properties of automata whose transition semigroups have some interesting properties, such as: to have a zero, left zero, right zero or bi-zero, to be a nilpotent extension of a rectangular band, left zero band or right zero band, to be nilpotent, etc. To describe the structure of these automata we use various well-known decomposition methods of automata theory, such as direct sum decompositions, subdirect and parallel decompositions, and extensions of automata. Automata that appear as the components in these decompositions belong to some well-known classes of automata. These are directable automata, introduced by P. H. Starke in [30] and J. Černý in [7], definite automata, defined first by S. C. Kleene in [20], and M. Perles, M. O. Rabin and E. Shamir in [25], reverse definite automata, introduced by J. A. Brzozowski in [5], and A. Ginzburg in [15], generalized definite automata, defined also by A. Ginzburg in [15], and nilpotent automata, that appeared first in the paper of L. N. Shevrin [29], and the book [14] by F. Gécseg and I. Peák. These automata were also studied by J. Černý, A. Piricka and B. Rosenauerová in [8], M. Ito and J. Duske in [19], J. E. Pin in [26], [27] and [28], M. Steinby in [31], and others. These types of automata were recently investigated by B. Imreh and M. Steinby in [18].

However, it is also necessary to introduce some new classes of automata. In Section 2 we define and study some new types of automata: generalized directable automata, trapped automata, one-trapped automata, locally directable automata and locally one-trapped automata. In Section 3 we introduce locally nilpotent and locally definite automata, and we connect them with nilpotent, definite, reverse definite and generalized automata. Relationships between these types of automata will be explained in Section 4, where the classes of these automata will be treated as generalized varieties. Note that the concept of an automaton 'belonging locally' to a given class of automata was introduced by M. Steinby in [32]

Automata considered throughout this paper will be automata without outputs in the sense of the definition from the book of F. Gécseg and I. Peák [14]. It is well known that automata without outputs, with the input alphabet X, can be considered as unary algebras of type indexed by X (we will say that they are of type X). This will be done throughout this paper. The notions such as a *congruence*, *subautomaton*, *generating set* etc., will have their usual algebraic meanings. In order to simplify notations, an automaton with the state set A will be also denoted by the same letter A. For any considered automaton A, its input alphabet will be denoted by X. In this paper we will aim our attention only to the case $|X| \geq 2$. The free monoid over X, i.e. the input monoid of A, is denoted by X^* and free semigroup over X is denoted by X^+. Under action of an input word $u \in X^*$, the automaton A goes from a state a into the state that will be denoted by au. For an arbitrary $k \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of all positive integers, we denote by X^k the set of all words having the length k, and by $X^{\geq k}$ the set of all words of the length at least k.

The *transition semigroup* $S = S(A)$ of an automaton A, in some origins called the *characteristic semigroup* of A, one can define in two equivalent ways. The first one is to define $S(A)$ as a semigroup consisting of all *transition mappings* on A,

by which we mean the mappings η_u, $u \in X^+$, defined by: $a\eta_u = au$, for $a \in A$. Another way is to define $S(A)$ to be the factor semigroup of the input semigroup X^+ with respect to the *Myhill congruence* μ on X^+ defined by: $(u,v) \in \mu$ if and only if $au = av$, for each $a \in A$. Note that $(u,v) \in \mu$ if and only if $\eta_u = \eta_v$. We will use the first way mostly.

Rees congruences, a famous notion of semigroup theory, have their analogues in many other theories. It appears that in automata theory they were first defined by I. Babcsányi in [2]. The *Rees congruence* on an automaton A determined by a subautomaton B of A is a congruence θ defined in the following way: For $a, b \in A$ we say that $(a,b) \in \theta$ if and only if either $a = b$ or $a, b \in B$ holds. The factor automaton A/θ is usually denoted by A/B, and it is called a *Rees factor automaton* of A with respect to B. If B is a subautomaton of an automaton A and the Rees factor automaton A/B is isomorphic to an automaton C, we say that A is an *extension* of an automaton B by an automaton C. Clearly, the automaton C can be viewed as an automaton obtained from A by contraction of B into a single element. In other words, C is isomorphic to the automaton D defined in the following way: $D = (A \setminus B) \cup \{a_0\}$, where a_0 does not belong to A, and the transitions in D are defined by

$$ax = \begin{cases} ax, & \text{as in } A, \text{ if } a, ax \in A \setminus B \\ a_0, & \text{if } a \in A \setminus B \text{ and } ax \notin A \setminus B, \text{ or } a = a_0 \end{cases}$$

We will usually identify the automata C and D.

Another notion imported from semigroup theory is the following: If there exists a homomorphism φ of an automaton onto its subautomaton B such that $a\varphi = a$, for each $a \in B$, then this homomorphism is called a *retraction* of A onto B and we say that A is a *retractive extension* of B by A/B.

An automaton A is a *direct sum* of its subautomata A_α, $\alpha \in Y$, if $A = \bigcup_{\alpha \in Y} A_\alpha$ and $A_\alpha \cap A_\beta = \varnothing$ for all $\alpha, \beta \in Y$ such that $\alpha \neq \beta$. The equivalence relation that correspond to this partition of A is a congruence and it is called a *direct sum congruence* on A. More information about general properties of direct sum decompositions of automata can be found in [10]. Finally, we say that an automaton A is a *parallel composition* of automata B and C if it can be embedded into their direct product.

For the notions and notations which are not explicitly defined here we refer to [6], [14] and [17].

2. Generalized directable automata

As it was announced in the introduction, we will investigate automata whose transition semigroups have some kinds of zeroes. Recall that an element e of a semigroup S is called a *left zero* of S if $es = e$, for each $s \in S$, a *right zero* of S if $se = e$, for each $s \in S$, and a *zero* of S, if it is both a left and a right zero of S. As a generalization of these notions we introduce the following notion: An element e of

a semigroup S will be called a *bi-zero* of S if $ese = e$, for each $s \in S$. First we describe semigroups having left, right or bi-zeroes.

Lemma 1. *A semigroup S has a bi-zero (resp. left zero, right zero) if and only if it is an ideal extension of a rectangular (resp. left zero, right zero) band.*

If e and f are bi-zeroes of S, then $esf = ef$, for each $s \in S$.

Proof. Suppose that S has a bi-zero. Let E denote the set of all bi-zeroes of S. For an arbitrary $e \in E$ we have $e^3 = e$ and $e^4 = ee^2e = e$, whence $e^2 = e$. Thus, E is a band, and clearly, it is a rectangular band. On the other hand, for $e \in E$ and $s, t \in S$ we have that $(es)t(es) = e(st)es = es$ and $(se)t(se) = se(ts)e = se$. Therefore, $es, se \in E$, so E is an ideal of S, which was to be proved.

Conversely, let S be an ideal extension of a rectangular band E. Assume arbitrary $e \in E$ and $s \in S$. Then $es \in E$, whence $ese = e(es)e = e$. Thus, e is a bi-zero of S.

The assertions concerning left and right zeroes can be proved similarly..

In the above notations, assume arbitrary $e, f \in E$ and $s \in S$. Then $sf \in E$ and $f = fef$, whence $esf = es(fef) = e(sf)ef = ef$. This completes the proof of the lemma. □

Using the previous one, we prove another lemma:

Lemma 2. *Let a semigroup S has a left (resp. right) zero. Then the set of all left (resp. right) zeroes of S coincides with the set of all bi-zeroes of S.*

If S has a zero, then it is unique and S does not have other left, right or bi-zeroes.

Proof. Let L and B denote the set of all left zeroes and the set of all bi-zeroes of S, respectively. Obviously, $L \subseteq B$. Assume an arbitrary $f \in B$. Then $f = fef$. But, by Lemma 1, L is an ideal of S, whence $f \in SLS \subseteq L$. Therefore, $L = B$.

The remaining assertions one proves similarly. □

Now we are passing from semigroups to automata. First we recall some known notions. An automaton A is called a *directable automaton* if there exists a word $u \in X^*$ such that $au = bu$, for all $a, b \in A$. Such word is called a *directable word* and the set of all directable words of A is denoted by $DW(A)$.

A state a of A is called a *trap* of A if $au = a$, for each $u \in X^*$, that is, if the set $\{a\}$ is a subautomaton of A [21]. If A has exactly one trap, it is called a *one-trap automaton* [3]. The set of all traps of A will be denoted by $Tr(A)$. An automaton whose each state is a trap is called a *discrete automaton* [13].

The first new notion that we introduce is the following: An automaton A will be called a *trapped automaton* if there exists a word $u \in X^*$ such that $au \in Tr(A)$, for each $a \in A$. Such word will be called a *trapping word*, and the set of all trapping words of A will be denoted by $TW(A)$. In other words, $u \in TW(A)$ if and only if $auv = au$, for all $a \in A$ and $v \in X^*$. We also define an automaton A to be a *one-trapped automaton* if it is trapped and has exactly one trap. It is not hard to

verify that A is a one-trapped automaton if and only if there exists $u \in X^*$ such that $auv = bu$, for all $a, b \in A$ and $v \in X^*$.

Third, we generalize directable automata as follows: An automaton A will be called a *locally directable automaton* if all monogenic subautomata of A are directable and they have common directing word. Here by a *monogenic subautomaton* we call a subautomaton generated by a single state (called also *cyclic*). The condition that all monogenic subautomata must have the same directing word is fulfilled in each finite automaton, that is, a finite automaton is locally directable if and only if all its monogenic subautomata are directable. Equivalently, A is locally directable if there exists $u \in X^*$ such that $avu = au$, for all $a \in A$ and $v \in X^*$. Such word will be called a *locally directing word*, and the set of all locally directing words of A will be denoted by $LDW(A)$.

Similarly, an automaton A will be called a *locally one-trapped automaton* if all monogenic subautomata of A are one-trapped automata and they have common trapping word. Such words will be called a *locally one-trapping word* of A and the set of all such words will be denoted by $LOTW(A)$. In other words, A is a locally one-trapped automaton if and only if there exists $u \in X^*$ such that $apuq = au$, for all $a \in A$ and $p, q \in X^*$. A finite automaton is locally one-trapped if and only if all its monogenic subautomata are one-trapped.

The fifth new notion that we introduce here is a common generalization of directable, locally directable and trapped automata. Namely, an automaton A is said to be a *generalized directable automaton* if there exists $u \in X^*$ such that $auvu = au$, for all $a \in A$ and $v \in X^*$. Such word will be called a *generalized directing word* of A. The set of all generalized directing words of an automaton A will be denoted by $GDW(A)$. We have chosen these names because an analogy with generalized definite automata, that will be considered in the next section.

The following lemma, that can be easily checked, establishes some relationships between these automata and the above considered semigroups.

Lemma 3. *Let A be an automaton and $u \in X^*$. Then $u \in GDW(A)$ (resp. $u \in TW(A)$, $u \in LDW(A)$, $u \in LOTW(A)$) if and only if η_u is a bi-zero (resp. left zero, right zero, zero) of $S(A)$.*

The next lemma is an immediate consequence of Lemmas 1, 2 and 3.

Lemma 4. *For an automaton A, $GDW(A)$, $TW(A)$ and $LDW(A)$ are ideals of X^*. Moreover, the following conditions hold:*

(1) *$TW(A) \neq \varnothing$ implies $TW(A) = GDW(A)$;*

(2) *$LDW(A) \neq \varnothing$ implies $LDW(A) = GDW(A)$;*

(3) *$LOTW(A) \neq \varnothing$ implies $LOTW(A) = LDW(A) = TW(A) = GDW(A)$;*

(4) *$TW(A) \neq \varnothing$ and $LDW(A) \neq \varnothing$ implies $LOTW(A) \neq \varnothing$.*

Now we are ready to prove one of the main theorems of the paper.

Theorem 1. *The following conditions on an automaton A are equivalent:*

(i) *$S(A)$ has a bi-zero;*

(ii) *A is an extension of a locally directable automaton by an one-trapped automaton;*

(iii) *A is a generalized directable automaton.*

Proof. (i)$\Leftrightarrow$(iii). This follows by Lemma 3.

(iii)$\Rightarrow$(ii). Let $B = \{au \mid a \in A, u \in GDW(A)\}$. Since $GDW(A)$ is an ideal of X^*, B is a subautomaton of A. Let a_0 be the trap of A/B which is the image of B under the natural homomorphism of A onto A/B. For arbitrary $a \in A \setminus B$ and $u \in GDW(A)$ we have that $au \in B$ in A, that is $au = a_0$ in A/B, so A/B is an one-trapped automaton.

Assume arbitrary $b \in B$, $v \in X^+$ and $w \in GDW(A)$. Then we have that $b = au$, for some $a \in A$ and $u \in GDW(A)$, and now $(bv)w = auvw = auw = bw$, by Lemma 1. This completes the proof of the implication (iii)$\Rightarrow$(ii).

(ii)$\Rightarrow$(iii). Let A be an extension of a locally directable automaton B by an one-trapped automaton A/B. Let $v \in LDW(B)$ and $u \in TW(A/B)$. Assume now arbitrary $a \in A$ and $w \in X^*$. Then $au \in B$, and since the subautomaton $S(au)$ of B generated by au is directable, with v as one of its directing words, and $au, auvwu \in S(au)$, then $auv = auvwuv$. Therefore, $uv \in GDW(A)$ and A is a generalized directable automaton. □

Locally directable automata, that appear in the above theorem, will be characterized by the next theorem.

Theorem 2. *The following conditions on an automaton A are equivalent:*

(i) *$S(A)$ has a right zero;*

(ii) *A is a direct sum of directable automata with the same directing word;*

(iii) *A is a locally directable automaton.*

If A is a finite automaton, then the condition (ii) *can be replaced by the following condition:*

(ii') *A is a direct sum of directable automata.*

Proof. (i)$\Leftrightarrow$(iii). This follows by Lemma 3.

(iii)$\Rightarrow$(ii). Assume an arbitrary $u \in LDW(A)$ and define a relation ϱ on A by: $(a, b) \in \varrho \Leftrightarrow au = bu$. Obviously, ϱ is an equivalence relation on A and $(av, a) \in \varrho$, for all $a \in A$ and $v \in X^*$. Therefore, by Lemma 3.1 of [10] we have that ϱ is a direct sum congruence on A.

Let B be an arbitrary ϱ-class of A. Assume arbitrary $a, b \in B$. Then $au = bu$, so B is a directable automaton, with u as one of its directing words. This completes the proof of the implication (iii)$\Rightarrow$(ii).

(ii)⇒(iii). Let A be a direct sum of directable automata A_α, $\alpha \in Y$, and let there exists a word $u \in X^*$ such that it is a directing word for all A_α, $\alpha \in Y$. Assume arbitrary $a \in A$ and $v \in X^*$. Then $a, av \in A_\alpha$, for some $\alpha \in Y$, and since A_α is directable and $u \in DW(A_\alpha)$, then $avu = au$, which was to be proved.

If A is finite and if (ii') holds, then A is a direct sum of finitely many directed automata $A_1, \ldots, A_k$, and if we assume an arbitrary $u_i \in DW(A_i)$, $i \in \{1, \ldots, k\}$, then $u = u_1 \cdots u_k \in DW(A_i)$, for each $i \in \{1, \ldots, k\}$, by Remark 3.2 of [18]. This completes the proof of the theorem. □

The next our goal is to characterize automata whose transition semigroups have left zeroes.

Theorem 3. *The following conditions on an automaton A are equivalent:*

(i) *$S(A)$ has a left zero;*

(ii) *A is an extension of a discrete automaton by an one-trapped automaton;*

(iii) *A is a trapped automaton.*

Proof. (i)⇔(iii). This follows by Lemma 3.

(iii)⇒(ii). By (i)⇔(iii) and Lemma 4, every trapped automaton A is a generalized directable automaton and $TW(A) = GDW(A)$. As was proved in (iii)⇒(ii) of Theorem 1, A is an extension of an automaton $B = \{au \mid a \in A, u \in GDW(A)\}$ by an one-trapped automaton, and since $GDW(A) = TW(A)$, then B is a discrete automaton.

(ii)⇒(iii). Let A be an extension of a discrete automaton B by an one-trapped automaton A/B and let $u \in Tr(A/B)$. Then for each $a \in A$ we have that $au \in B = Tr(A)$, so we have proved that A is trapped. □

We will finish this section considering automata whose transition semigroups have a zero.

Theorem 4. *The following conditions on an automaton A are equivalent:*

(i) *$S(A)$ has a zero;*

(ii) *A is a retractive extension of a discrete automaton by an one-trapped automaton;*

(iii) *A is a direct sum of one-trapped automata with the same trapping word;*

(iv) *A is a subdirect product of a discrete automaton and an one-trapped automaton;*

(v) *A is a parallel composition of a discrete automaton and an one-trapped automaton;*

(vi) *A is a locally one-trapped automaton;*

If A is a finite automaton, then the condition (iii) *can be replaced by the following condition:*

(iii') *A is a direct sum of one-trapped automata.*

Proof. (i)$\Leftrightarrow$(vi). This follows by Lemma 3.

(vi)$\Rightarrow$(ii). Let A be a locally one-trapped automaton. Assume an arbitrary $u \in LOTW(A)$. Then A is trapped, and by Theorem 3, A is an extension of a discrete automaton $B = Tr(A)$ by a one-trapped automaton A/B. Define a mapping φ of A into B by: for $a \in A$, $a\varphi = au$. Since $au \in Tr(A)$ and A is locally one-trapped, for each $v \in X^*$ we have that $(av)\varphi = avu = au = auv = (a\varphi)v$, so φ is a homomorphism. On the other hand, if $a \in B$, then it is a trap and $a\varphi = au = a$. Therefore, φ is a retraction of A onto B, which was to be proved.

(ii)$\Rightarrow$(iii). Let A be a retractive extension of a discrete automaton B by a one-trapped automaton A/B. Let φ be a retraction of A onto B and let u be an arbitrary trapping word of A/B. For $b \in B$, let $A_b = b\varphi^{-1}$. Since an inverse homomorphic image of a subautomaton is also a subautomaton, then A_b, $b \in B$, are subautomata of A and A is a direct sum of these automata. Clearly, b is the unique trap of A_b and u is a trapping word of A_b. Thus, we have proved (iii).

(iii)$\Rightarrow$(iv). Let A be a direct sum of one-trapped automata A_α, $\alpha \in Y$, that have the same trapping word u. Let σ denote the corresponding direct sum congruence on A. As we know, A/σ is a discrete automaton. On the other hand, $B = Tr(A)$ is a subautomaton of A. Let ϱ denote the Rees congruence on A determined by B. Obviously, A/ϱ is an one-trapped automaton, with u as one of its trapping words. Finally, $\sigma \cap \varrho = \Delta$, since each σ-class contains exactly one trap of A. Here Δ denotes the equality relation on A. Therefore, A is a subdirect product of A/σ and A/ϱ, so we have proved (iv).

(iv)$\Rightarrow$(v). This implication is obvious.

(v)$\Rightarrow$(vi). Let A be a parallel composition of a discrete automaton B and a one-trapped automaton C. Let ϕ be an embedding of A into $B \times C$, and let u be an arbitrary trapping word of C. Assume arbitrary $a \in A$ and $p, q \in X^*$. Then $a\phi = (b, c)$ for some $b \in B$ and $c \in C$, so $(apuq)\phi = (a\phi)puq = (bpuq, cpuq) = (b, cu) = (bu, cu) = (a\phi)u = (au)\phi$, whence $apuq = au$, which was to be proved.

If A is finite and if (iii') holds, then A is a direct sum of finitely many one-trapped automata $A_1, \ldots, A_k$, and if we assume an arbitrary $u_i \in TW(A_i)$, $i \in \{1, \ldots, k\}$, then $u = u_1 \cdots u_k \in TW(A_i)$, for each $i \in \{1, \ldots, k\}$, by Lemma 4. This completes the proof of the theorem. □

3. Generalized definite automata

In this section we study the class of generalized definite automata and some of its well-known subclasses, from the aspect of properties of transition semigroups of automata belonging to these classes.

First we recall some known definitions. An automaton A is called a *definite automaton* if there exists $k \in \mathbb{N}$ such that $au = bu$, for all $a, b \in A$ and $u \in X^{\geq k}$, or

equivalently, if $X^{\geq k} \subseteq DW(A)$, for some $k \in \mathbb{N}$. The smallest number having this property is called the *degree of definiteness* of A. Similarly, A is called a *reverse definite automaton* if there exists $k \in \mathbb{N}$ such that $auv = au$, for all $a \in A$, $u \in X^{\geq k}$ and $v \in X^*$, that is, if $X^{\geq k} \subseteq TW(A)$, for some $k \in \mathbb{N}$. The smallest number having this property is called the *degree of reverse definiteness* of A.

An automaton A is called a *nilpotent automaton* if it has a unique trap and there exists $k \in \mathbb{N}$ such that each word $u \in X^{\geq k}$ is a trapping word. In other words, A is nilpotent if and only if there exists $k \in \mathbb{N}$ such that $auv = bu$, for all $a, b \in A$, $u \in X^{\geq k}$ and $v \in X^*$. The smallest number having this property is called the *degree of nilpotency* of A. An extension A of an automaton B will be called a *nilpotent extension* of B if the factor automaton A/B is nilpotent. Clearly, A is a nilpotent extension of B if and only if there exists $k \in \mathbb{N}$ such that $au \in B$ for all $a \in A$ and $u \in X^{\geq k}$.

As in the previous section, we give some new definitions regarding some "local" properties of automata. An automaton A will be called *locally definite* if all its monogenic subautomata are definite and their degrees of definiteness are bounded. For finite automata the second condition is obviously fulfilled and it can be omitted. Equivalently, A is locally definite if and only if there exists $k \in \mathbb{N}$ such that $avu = au$, for all $a \in A$, $v \in X^*$ and $u \in X^{\geq k}$.

Similarly, A is said to be *locally nilpotent* if all its monogenic subautomata are nilpotent and their degrees of nilpotency are bounded. As in the previous case, the second condition can be omitted for finite automata. In other words, A is locally nilpotent if and only if there exists $k \in \mathbb{N}$ such that $apuq = au$, for all $a \in A$, $p, q \in X^*$ and $u \in X^{\geq k}$.

Finally, by a *generalized definite automaton* we mean an automaton for which there exist $k, m \in \mathbb{N}$ such that $aupv = auqv$, for all $a \in A$, $p, q \in X^*$, $u \in X^{\geq k}$ and $v \in X^{\geq m}$. These automata are described by the following theorem:

Theorem 5. *The following conditions on an automaton A are equivalent:*

(i) *$S(A)$ is a nilpotent extension of a rectangular band;*

(ii) *A is a nilpotent extension of a locally definite automaton;*

(iii) *A is a generalized definite automaton;*

(iv) $(\exists k \in \mathbb{N})(\forall u \in X^{\geq k})(\forall a \in A)(\forall v \in X^*)\ auvu = au$.

Proof. (i)⇒(iii). Let S be a nilpotent extension of a rectangular band E, i.e. $S^k = E$, for some $k \in \mathbb{N}$. Assume arbitrary $u, v \in X^{\geq k}$, $p, q \in X^*$ and $a \in A$. Then $\eta_u, \eta_v \in E$, whence $\eta_{upv} = \eta_u \eta_p \eta_v = \eta_u \eta_v = \eta_u \eta_q \eta_v = \eta_{uqv}$, whence $aupv = auqv$, which was to be proved.

(iii)⇒(iv). If A is generalized definite, then there exist $m, n \in \mathbb{N}$ such that $aupv = auqv$, for all $a \in A$, $p, q \in X^*$, $u \in X^{\geq m}$ and $v \in X^{\geq n}$. Let $k = m + n$, $w \in X^{\geq k}$, $a \in A$ and $p \in X^*$. Then $w = uv$, for some $u \in X^{\geq m}$ and $v \in X^{\geq n}$, whence $awpw = au(vpu)v = auv = aw$. Therefore, (iv) holds.

(iv)⇒(i). We see that A is a generalized directable automaton, so S is an ideal extension of a rectangular band E consisting of all bi-zeroes of S. Moreover, the condition (iv) means that $X^{\geq k} \subseteq GDW(A)$, for some $k \in \mathbb{N}$, so we conclude the following: if $s \in S^k$, then $s = \eta_u$, where u can be chosen to be in $X^{\geq k}$, that is, to be in $GDW(A)$. Now by Lemmas 1 and 3 we have that $s = \eta_u \in E$. Therefore, $S^k = E$, which was to be proved.

(iv)⇒(ii). Since A is a generalized directable automaton, then by Theorem 1, it is an extension of a locally directable automaton $B = \{au \mid a \in A,\ u \in GDW(A)\}$ by a one-trapped automaton A/B. But, by (iv) we have that $X^{\geq k} \subseteq GDW(A)$, for some $k \in \mathbb{N}$, so $au \in B$, for any $a \in A$ and $u \in X^{\geq k}$. Therefore, A/B is a nilpotent automaton. Assume arbitrary $b \in B$, $u \in X^{\geq k}$ and $v \in X^*$. Then $b = aw$, for some $w \in X^{\geq k}$, so by Lemmas 1 and 3 it follows that $bvu = awvu = awu = bu$, since $u, w \in X^{\geq k} \subseteq GDW(A)$. Thus, B is locally definite.

(ii)⇒(iii). Let A be a nilpotent extension of a locally definite automaton B. Then there exists $k \in \mathbb{N}$ such that $au \in B$, for all $a \in A$ and $u \in X^{\geq k}$, and there exists $m \in \mathbb{N}$ such that $bwv = bv$, for all $b \in B$, $w \in X^*$ and $v \in X^{\geq m}$. Assume now arbitrary $u \in X^{\geq k}$, $v \in X^{\geq m}$, $a \in A$ and $p, q \in X^*$. Then $au \in B$ yields $aupv = (au)pv = (au)v = (au)qv = auqv$. Therefore, A is generalized definite. □

The condition (iv) will be used here as a simpler definition of the generalized definiteness. Note again that this condition means that $X^{\geq k} \subseteq GDW(A)$, for some $k \in \mathbb{N}$.

Next we intend to describe structure of locally definite automata that appear in the preceding theorem.

Theorem 6. *The following conditions on an automaton A are equivalent:*

(i) *$S(A)$ is a nilpotent extension of a right zero band;*

(ii) *A is a direct sum of definite automata with bounded degrees of definiteness;*

(iii) *A is a locally definite automaton.*

If A is a finite automaton, then the condition (ii) *can be replaced by the following condition:*

(ii') *A is a direct sum of definite automata.*

Proof. (i)⇒(iii). Let S be a nilpotent extension of a right zero band E. Assume $k \in \mathbb{N}$ such that $S^k = E$. In view of Lemmas 1 and 3, $S^k = E$ implies that $X^{\geq k} \subseteq LDW(A)$, which is clearly equivalent to the condition (iii).

(iii)⇒(i). Clearly, A is generalized definite, so by Theorem 5 it follows that S is a nilpotent extension of a rectangular band E which consists of all bi-zeroes of S. On the other hand, A is locally directable, so by Theorem 2 and Lemmas 1 and 2 we have that E is also the set of all right zeroes of S, i.e. it is a right zero band.

(iii)⇒(ii). Assume $k \in \mathbb{N}$ such that $avu = au$, for all $a \in A$, $u \in X^{\geq k}$ and $v \in X^*$. Let a relation ϱ on A be defined by: $(a, b) \in \varrho \Leftrightarrow (\forall u \in X^{\geq k})\ au = bu$.

It is easy to see that ϱ is an equivalence relation on A. On the other hand, the definition of local definiteness implies that ϱ is a direct sum congruence on A. Let B be an arbitrary ϱ-class of A and $a, b \in B$. Then $au = bu$, for each $u \in X^{\geq k}$, so B is a definite automaton whose degree of definiteness does not exceed k. Therefore, (ii) holds.

(ii)$\Rightarrow$(iii). Let A be a direct sum of definite automata A_α, $\alpha \in Y$, and let k be a bound of their degrees of definiteness. Assume arbitrary $a \in A$, $v \in X^*$ and $u \in X^{\geq k}$. Then $a, av \in A_\alpha$, for some $\alpha \in Y$, so $avu = au$, since $u \in DW(A_\alpha)$. This proves (iii).

As in the proof of Theorem 2 we show that (ii) is equivalent to (ii') for all finite automata. □

An automaton A is called a *reset automaton* if it is a definite automaton with the degree of definiteness equal to 1, that is if $ax = bx$, for all $a, b \in A$ and $x \in X$. If all monogenic subautomata of A are reset, we will say that A is a *locally reset automaton*. In other words, A is locally reset if and only if $aux = ax$, for all $a \in A$, $x \in X$ and $u \in X^*$. As an immediate consequence of the previous theorem we have the following:

Corollary 1. *The following conditions on an automaton A are equivalent:*

(i) *$S(A)$ is a right zero band;*

(ii) *A is a direct sum of reset automata;*

(iii) *A is a locally reset automaton.*

Next we consider automata whose transition semigroups are nilpotent extensions of left zero bands.

Theorem 7. *The following conditions on an automaton A are equivalent:*

(i) *$S(A)$ is a nilpotent extension of a left zero band;*

(ii) *A is a nilpotent extension of a discrete automaton;*

(iii) *A is a reverse definite automaton.*

Proof. (i)$\Leftrightarrow$(iii). Assume $k \in \mathbb{N}$ such that $S^k = E$ is a left zero band. Then $s \in E$ if and only if $s = \eta_u$, for some $u \in X^{\geq k}$, and, on the other hand, $\eta_u \in E$ if and only if $u \in TW(A)$. Therefore, S^k is a left zero band, for some $k \in \mathbb{N}$, if and only if A is reverse definite.

(iii)$\Rightarrow$(ii). By Theorem 3, A is an extension of a discrete automaton B by a one-trapped automaton A/B, and then $B = Tr(A)$. On the other hand, by (iii) it follows that there exists $k \in \mathbb{N}$ such that $au \in B$, for each $u \in X^{\geq k}$. Thus, A/B is a nilpotent automaton, which was to be proved.

(ii)$\Rightarrow$(iii). Let A be a nilpotent extension of a discrete automaton B. Clearly, $B = Tr(A)$. Let k be the degree of nilpotency of A/B, and assume arbitrary $u \in X^{\geq k}$, $a \in A$ and $v \in X^*$. Then $au \in B$, whence $auv = au$. Thus, A is reverse definite. □

Theorem 8. *The following conditions on an automaton A are equivalent:*

(i) *$S(A)$ is a nilpotent semigroup;*

(ii) *A is a retractive nilpotent extension of a discrete automaton;*

(iii) *A is a direct sum of nilpotent automata with bounded degrees of nilpotency;*

(iv) *A is a subdirect product of a discrete automaton and a nilpotent automaton;*

(v) *A is a parallel composition of a discrete automaton and a nilpotent automaton;*

(vi) *A is a locally nilpotent automaton;*

If A is a finite automaton, then the condition (iii) *can be replaced by the following condition:*

(iii') *A is a direct sum of nilpotent automata.*

Proof. Note that the equivalence of conditions (i) and (iii) was discovered by L. N. Shevrin in [29], and one proof of this assertion can be found in the book of F. Gécseg and I. Peák [14]. However, here we will give another proof of this assertion.

(i)$\Leftrightarrow$(vi). We see that A is locally nilpotent if and only if $X^{\geq k} \subseteq LOTW(A)$, for some $k \in \mathbb{N}$. But, this holds if and only if S has a zero 0 and $S^k = \{0\}$, for some $k \in \mathbb{N}$, by Lemma 3.

(vi)$\Rightarrow$(ii). By Theorem 4, A is a retractive extension of a discrete automaton B by an one-trapped automaton. On the other hand, by Theorem 7, A is a nilpotent extension of a discrete automaton C. Clearly, $B = C$, so (ii) is proved.

(ii)$\Rightarrow$(iii). This one proves similarly as the corresponding part of the proof of Theorem 4.

(iii)$\Rightarrow$(iv). Let A be a direct sum of nilpotent automata A_α, $\alpha \in Y$, and let k be a bound of the degrees of nilpotency of the summands A_α, $\alpha \in Y$. By the proof of Theorem 4, A is a subdirect product of a discrete automaton A/σ and an one-trapped automaton A/ϱ, where σ and ϱ are congruences on A defined as in the proof of Theorem 4. It is not hard to check that A/ϱ is a nilpotent automaton with the degree of nilpotency which does not exceed k.

(iv)$\Rightarrow$(v). This is obvious.

(v)$\Rightarrow$(vi). Let A be a parallel composition of a discrete automaton B and a nilpotent automaton C. Then $X^{\geq k} \subseteq LOTW(C)$, for some $k \in \mathbb{N}$, and if assume arbitrary $u \in X^{\geq k}$, $a \in A$ and $p, q \in X^*$, as in the proof of Theorem 4 we obtain that $apuq = au$, which was to be proved.

The rest of the proof can be proved similarly as the related parts of the proof of Theorems 4 and 6. □

Note that finite semigroups which are nilpotent extensions of rectangular bands are known as *locally trivial* semigroups. Languages that correspond to these semigroups, in the sense of the Eilenberg's theorem, were characterized in the book [28] by J. E. Pin. Languages that correspond to finite nilpotent semigroups, and finite semigroups which are nilpotent extensions of left and right zero bands, were also described in this book.

4. Characterizations through generalized varieties

Treatment of X-automata as unary algebras of type X gives possibility to study varieties of X-automata and certain their generalizations, such as generalized varieties and pseudo-varieties. In this section we use this possibility to characterize the classes considered in the previous two sections as generalized varieties of automata.

A class K of X-automata is called a *variety* if it is closed under homomorphisms, subautomata and direct products, a *generalized variety*, if it is closed under homomorphisms, subautomata, finite direct products and arbitrary direct powers, and it is called a *pseudo-variety* if it consists only of finite automata and it is closed under homomorphisms, subautomata and finite direct products. As was proved by C. J. Ash in [1], K is a generalized variety if and only if it is a directed union of varieties, and it is a pseudo-variety if and only if it is the intersection of some generalized variety and the pseudo-variety of all finite X-automata. Generalized varieties will be here usually denoted by bold face letters. For a generalized variety $\mathbf{K}$, the corresponding pseudo-variety, consisting of all finite automata from $\mathbf{K}$, will be denoted by $\underline{\mathbf{K}}$.

As known, a class of algebras of a given type τ is a variety if and only if it can be equationally defined, that is, if it is the class of all algebras of type τ that satisfy a given set of identities of type τ. It is also known that this set of identities can be chosen so that at most countably many variables occur in them. For automata, this set of variables can be obviously reduced to at most two variables. So we will consider identities of type X in at most two variables, that is, the identities of the form $gu = hv$ or $gu = gv$, where $u, v \in X^*$ and g and h are variables that take their values in the set of states of an automaton. If a family $\{gu_i = hv_i\}_{i \in I}$ of identities of type X is given, then $[gu_i = hv_i \mid i \in I]$ will denote the variety of X-automata determined by this family of identities.

We introduce the following notations:

Notation	Class of automata	Notation	Class of automata
GDir	generalized directable	**GDef**	generalized definite
LDir	locally directable	**LDef**	locally definite
Dir	directable	**Def**	definite
Trap	trapped	**RDef**	reverse definite
LOTrap	locally one-trapped	**LNilp**	locally nilpotent
OTrap	one-trapped	**Nilp**	nilpotent
D	discrete	**O**	trivial

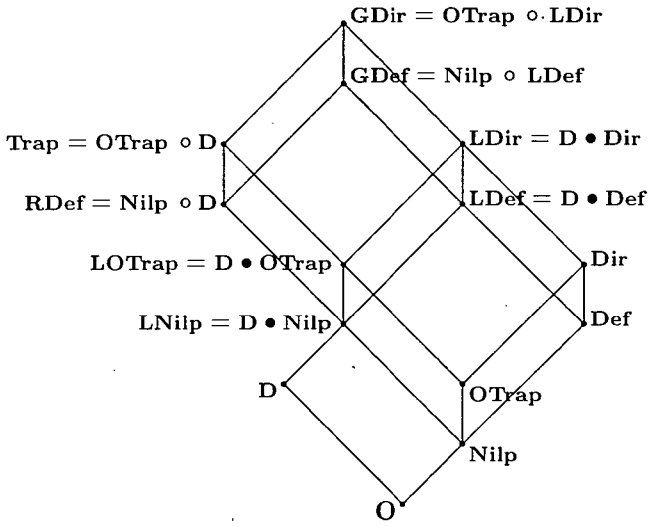
Table 1

Let K_1 and K_2 be two classes of X-automata. Then their *Mal'cev product* $K_1 \circ K_2$ is defined as the class of all X-automata A such that there exists a congruence ϱ on A so that A/ϱ belongs to K_2 and every ϱ-class which is a subautomaton of A belongs to K_1. For example, $\mathbf{OTrap} \circ K$ is the class of all extensions of automata from K by one-trapped automata, and $\mathbf{D} \circ K$ denotes the class of all automata that

are direct sums of automata from K. Especially, $\mathbf{D} \bullet \mathbf{Dir}$ will denote all direct sums of directable automata with the same directing word, $\mathbf{D} \bullet \mathbf{Def}$ will denote all direct sums of definite automata with bounded degrees of definiteness, $\mathbf{D} \bullet \mathbf{OTrap}$ will denote all direct sums of one-trapped automata with the same trapping word, and $\mathbf{D} \bullet \mathbf{Nilp}$ will denote all direct sums of nilpotent automata with bounded degrees of nilpotency.

Now we are ready to prove the following theorem:

Theorem 9. *The classes defined in Table* 1 *are pairwise different generalized varieties and the following figure represents their inclusion diagram:*



Moreover, they form a semilattice under the set intersection.

Proof. Clearly, $\mathbf{D}$ and $\mathbf{O}$ are varieties. Other classes can be represented in the following way:

$$\mathbf{GDir} = \bigcup_{u \in X^*} [guwu = gu \mid w \in X^*], \quad \mathbf{GDef} = \bigcup_{k \in \mathbb{N}} [guwu = gu \mid u \in X^{\geq k}, w \in X^*],$$

$$\mathbf{Dir} = \bigcup_{u \in X^*} [gu = hu], \quad \mathbf{Def} = \bigcup_{k \in \mathbb{N}} [gu = hu \mid u \in X^{\geq k}],$$

$$\mathbf{Trap} = \bigcup_{u \in X^*} [guw = gu \mid w \in X^*], \quad \mathbf{RDef} = \bigcup_{k \in \mathbb{N}} [guw = gu \mid u \in X^{\geq k}, w \in X^*],$$

$$\mathbf{OTrap} = \bigcup_{u \in X^*} [guw = hu \mid w \in X^*], \quad \mathbf{Nilp} = \bigcup_{k \in \mathbb{N}} [guw = hu \mid u \in X^{\geq k}, w \in X^*],$$

$$\mathbf{LDir} = \bigcup_{u \in X^*} [gwu = gu \mid w \in X^*], \quad \mathbf{LDef} = \bigcup_{k \in \mathbb{N}} [gwu = gu \mid u \in X^{\geq k}, w \in X^*],$$

$$\mathbf{LOTrap} = \bigcup_{u \in X^*} [gpuq = gu \mid p, q \in X^*], \quad \mathbf{LNilp} = \bigcup_{k \in \mathbb{N}} [gpuq = gu \mid u \in X^{\geq k}, p, q \in X^*].$$

On the other hand, since $GDW(A)$, $LDW(A)$ and $TW(A)$ and $LOTW(A)$, if they are non-empty, are ideals of X^*, for every X-automaton A, we have:

$$[guwu = gu \mid w \in X^*],\ [gvwv = gv \mid w \in X^*] \subseteq [guvwuv = guv \mid w \in X^*],$$
$$[gu = hu],\ [gv = hv] \subseteq [guv = huv],$$
$$[guw = gu \mid w \in X^*],\ [gvw = gv \mid w \in X^*] \subseteq [guvw = guv \mid w \in X^*],$$
$$[guw = hu \mid w \in X^*],\ [gvw = hv \mid w \in X^*] \subseteq [guvw = huv \mid w \in X^*],$$
$$[gwu = gu \mid w \in X^*],\ [gwv = gv \mid w \in X^*] \subseteq [gwuv = guv \mid w \in X^*],$$
$$[gpuq = gu \mid p, q \in X^*],\ [gpvq = gv \mid p, q \in X^*] \subseteq [gpuvq = guv \mid p, q \in X^*],$$

and for $m, k \in \mathbb{N}$, $m \geq k$ implies

$$[guwu = gu \mid u \in X^{\geq k},\ w \in X^*] \subseteq [guwu = gu \mid u \in X^{\geq m},\ w \in X^*],$$
$$[gu = hu \mid u \in X^{\geq k}] \subseteq [gu = hu \mid u \in X^{\geq m}],$$
$$[guw = gu \mid u \in X^{\geq k},\ w \in X^*] \subseteq [guw = gu \mid u \in X^{\geq m},\ w \in X^*],$$
$$[guw = hu \mid u \in X^{\geq k},\ w \in X^*] \subseteq [guw = hu \mid u \in X^{\geq m},\ w \in X^*],$$
$$[gwu = gu \mid u \in X^{\geq k},\ w \in X^*] \subseteq [gwu = gu \mid u \in X^{\geq m},\ w \in X^*],$$
$$[gpuq = gu \mid u \in X^{\geq k},\ p, q \in X^*] \subseteq [gpuq = gu \mid u \in X^{\geq m},\ p, q \in X^*].$$

Therefore, each of the above given unions is directed, that is, each of the given classes is a directed union of varieties, so by Theorem 1 of [1], they are generalized varieties.

It is not hard to verify that the above figure represents the inclusion diagram of the considered classes. This follows by the given representation of these generalized varieties and by Theorems 1–8. We will give some examples that verify that these inclusions are proper.

Let the input alphabet X be represented in the form $X = X_1 \cup X_2$, where $X_1 \neq \varnothing$, $X_2 \neq \varnothing$ and $X_1 \cap X_2 = \varnothing$. This is possible since the automata with the one-element input alphabets are out of consideration. Consider the automata constructed by the following figures:

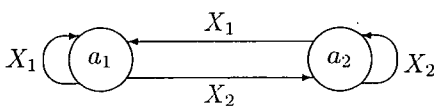


Fig. 1

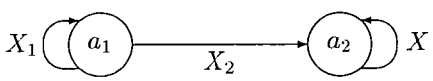


Fig. 2

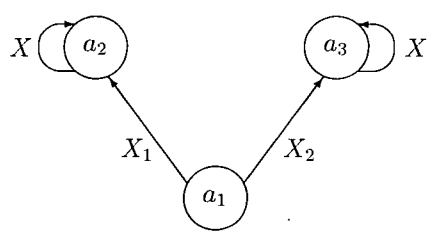


Fig. 3

The automaton from Fig. 1 is a two-element reset automaton and it belongs to

Def \ **Trap**, that yields the inclusions

$$\begin{array}{lll} \mathbf{Nilp} \subset \mathbf{Def}, & \mathbf{LNilp} \subset \mathbf{LDef}, & \mathbf{RDef} \subset \mathbf{GDef} \\ \mathbf{OTrap} \subset \mathbf{Dir}, & \mathbf{LOTrap} \subset \mathbf{LDir}, & \mathbf{Trap} \subset \mathbf{GDir}. \end{array}$$

The automaton given by Fig. 2 belongs to **OTrap** \ **GDef**, whence it follows that

$$\begin{array}{lll} \mathbf{Nilp} \subset \mathbf{OTrap}, & \mathbf{LNilp} \subset \mathbf{LOTrap}, & \mathbf{RDef} \subset \mathbf{Trap} \\ \mathbf{Def} \subset \mathbf{Dir}, & \mathbf{LDef} \subset \mathbf{LDir}, & \mathbf{GDef} \subset \mathbf{GDir}. \end{array}$$

The third automaton, defined by Fig. 3, belongs to **RDef** \ **LDir**, so we conclude that

$$\mathbf{LNilp} \subset \mathbf{RDef}, \quad \mathbf{LOTrap} \subset \mathbf{Trap}, \quad \mathbf{LDef} \subset \mathbf{GDef}, \quad \mathbf{LDir} \subset \mathbf{GDir}.$$

Assume an arbitrary $B \in \mathbf{Nilp}$. Let A be the direct sum of at least two isomorphic copies of B. Then A belongs to **LNilp** \ **Dir**, and this yields the inclusions

$$\mathbf{Nilp} \subset \mathbf{LNilp}, \quad \mathbf{OTrap} \subset \mathbf{LOTrap}, \quad \mathbf{Def} \subset \mathbf{LDef}, \quad \mathbf{Dir} \subset \mathbf{LDir}.$$

The inclusions $\mathbf{O} \subset \mathbf{Nilp}$, $\mathbf{O} \subset \mathbf{D}$ and $\mathbf{D} \subset \mathbf{LNilp}$ are obvious. Therefore, we have proved that all classes given in the above figure are different.

Further, assume $A \in \mathbf{Trap} \cap \mathbf{Dir}$. Then $TW(A) \neq \varnothing$ and $LDW(A) \neq \varnothing$, so $LOTW(A) \neq \varnothing$, by Lemma 4, whence $A \in \mathbf{LOTrap} = \mathbf{D} \bullet \mathbf{OTrap}$. But, A is direct sum indecomposable, since $A \in \mathbf{Dir}$, so $A \in \mathbf{OTrap}$. Thus, $\mathbf{Trap} \cap \mathbf{Dir} = \mathbf{OTrap}$. By this it also follows that $\mathbf{K} \cap \mathbf{Dir} = \mathbf{OTrap}$, for each $\mathbf{K}$ from the figure such that $\mathbf{OTrap} \subseteq \mathbf{K} \subseteq \mathbf{Trap}$.

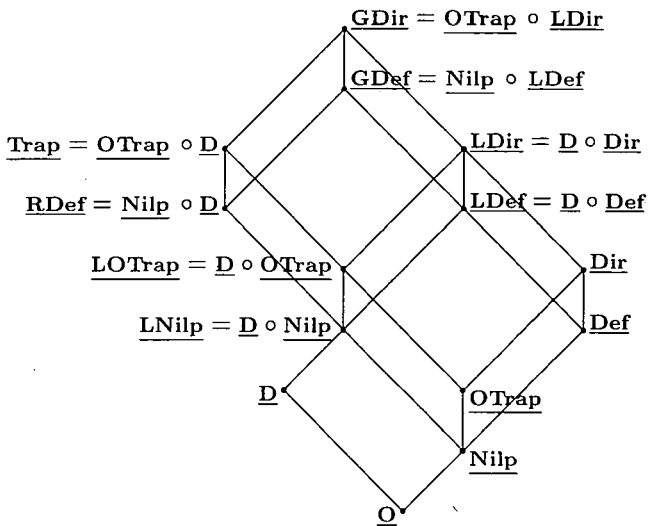
Let $A \in \mathbf{Trap} \cap \mathbf{Def}$. Then we also have $A \in \mathbf{OTrap}$ and $LOTW(A) = LDW(A) \neq \varnothing$. On the other hand, $A \in \mathbf{Def}$ implies that $X^{\geq k} \subseteq LDW(A)$, for some $k \in \mathbb{N}$, and now $X^{\geq k} \subseteq LOTW(A)$, whence $A \in \mathbf{Nilp}$. Thus, we have proved $\mathbf{Trap} \cap \mathbf{Dir} = \mathbf{Nilp}$, and this implies that $\mathbf{K} \cap \mathbf{Def} = \mathbf{Nilp}$, for each $\mathbf{K}$ from the figure such that $\mathbf{Nilp} \subseteq \mathbf{K} \subseteq \mathbf{Trap}$.

In the same way we prove that $\mathbf{Trap} \cap \mathbf{LDir} = \mathbf{LOTrap}$ and $\mathbf{Trap} \cap \mathbf{LDef} = \mathbf{LNilp}$, that implies that $\mathbf{K} \cap \mathbf{LDef} = \mathbf{LNilp}$, for each $\mathbf{K}$ from the figure such that $\mathbf{LNilp} \subseteq \mathbf{K} \subseteq \mathbf{Trap}$. Finally, it is clear that $\mathbf{D} \cap \mathbf{K} = \mathbf{O}$, for every $\mathbf{K}$ from the figure such that $\mathbf{K} \subseteq \mathbf{Dir}$.

Therefore, the above diagram represents a semilattice under the set intersection. This completes the proof of the theorem. □

An immediate consequence of the previous theorem is its analogue concerning related pseudo-varieties.

Corollary 2. *The classes given in the following figure are pairwise different pseudo-varieties and the figure represents their inclusion diagram:*



Remark 1. Previously we considered only automata with at least two input letters. In the case of *autonomous automata*, i.e. the automata whose input alphabet is one-element, we have that only the classes **Nilp**, **LNilp**, **O** and **D** are different since the transition semigroup of an autonomous automaton is monogenic.

References

[1] C. J. Ash,, Pseudovarieties, generalized varieties and similarly described classes, *J. Algebra* **92** (1985), 104–115.

[2] I. Babcsányi, Rees automaták, *Matematikai Lapok* **29** (1977-81), no. 1–3, 139–148 (in Hungarian).

[3] I. Babcsányi and A. Nagy, Right-group type automata, *Acta Cybernetica* **12** (1995), 131–136.

[4] S. Bogdanović and M. Ćirić, A note on congruences on algebras, in *Proc. of II Math. Conf. in Priština 1996*, Lj. D. Kočinac ed., Priština, 1997, pp. 67–72.

[5] J. A. Brzozowski, Canonical regular expressions and minimal state graphs for definite events, *Proc. Symp. Math. Theory of Automata*, Microwave Research Inst. Symp. Ser. **12** (Brooklyn, 1963), New York, 1963, pp. 529–561.

[6] S. Burris and H. P. Sankappanávar, *A course in universal algebra*, Springer–Verlag, New York, 1981.

[7] J. Černý, Poznámka k homogénym experimentom s konecinými automatami, *Mat.-fyz. cas. SAV* **14** (1964), 208–215.

[8] J. Černý, A. Pirická and B. Rosenauerová, On directable automata, *Kybernetika* **7** (1971), no. 4, 289–298.

[9] M. Ćirić and S. Bogdanović, Posets of C-congruences, *Algebra Universalis* **36** (1996), 423–424.

[10] M. Ćirić and S. Bogdanović, Lattices of subautomata and direct sum decompositions of automata, Algebra Colloq. (to appear).

[11] M. Ćirić, S. Bogdanović and T. Petković, The lattice of positive quasi-orders on an automaton, *Facta Universitatis (Niš) Ser. Math. Inform.* **11**, (1996), 143–156.

[12] M. Ćirić, S. Bogdanović and T. Petković, The lattice of subautomata of an automaton (a brief survey), *Proceedings of the Conference LIRA '97*, (to appear).

[13] P. Dömösi, On temporal products of automata, *Papers on Automata and Languages X*, Dept. of Math. Karl Marx Univ. of Economics, Budapest, 1988-1, pp. 49–62.

[14] F. Gécseg and I. Peák, *Algebraic Theory of Automata*, Akadémiai Kiadó, Budapest, 1972.

[15] A. Ginzburg, About some properties of definite, reverse definite and related automata, *IEEE Trans. Electronic Computers* **EC-15** (1966), 809–810.

[16] V. M. Glushkov, Abstract automata and partitions of free semigroups, *DAN SSSR* **136** (1961), 765–767 (in Russian).

[17] J. M. Howie, *Fundamentals of Semigroup Theory*, London Math. Soc. monographs, New Series, Clarendon Press, Oxford, 1995.

[18] B. Imreh and M. Steinby, Some remarks on directable automata, *Acta Cybernetica* **12**, (1995), no. 1, 23–35.

[19] M. Ito and J. Duske, On cofinal and definite automata, *Acta Cybernetica* **6** (1983), no. 2, 181–189.

[20] S. C. Kleene, Representation of events in nerve nets and finite automata, *Automata Studies*, Princeton University Press, Princeton, N.J., 1956, pp. 3–41.

[21] W. Lex and R. Wiegandt, Torsion theory for acts, *Studia Sci. Math. Hungar.* **16** (1981), 263–280.

[22] A. Nagy, Boolean type retractable automata with traps, *Acta Cybernetica* **10** (1991), no. 1–2, 53–64.

[23] I. Peák, Automata and semigroups I, *Acta Sci. Math. (Szeged)* **25** (1964), 193–201 (in Russian).

[24] I. Peák, Automata and semigroups II, *Acta Sci. Math. (Szeged)* **26** (1965), 49–54 (in Russian).

[25] M. Perles, M. O. Rabin and E. Shamir, The theory of definite automata, *IEEE Trans. Electronic Computers* **EC-12** (1963), 233–243.

[26] J. E. Pin, Sur les mots synchronisants dans un automata fini, *Elektron. Inform. Verarb. u. Kybernetik, EIK* **14** (1978), 297–303.

[27] J. E. Pin, Sur un cas particulier de la conjecture de Cerny, *Automata, langagues and programming*, ICALP'79 (Proc. Coll., Udine, 1979), Lect. Notes. Comp. Sci. **62**, Springer-Verlag, Berlin, 1979, pp. 345–352.

[28] J. E. Pin, *Variétés de langages formels*, Masson, Paris, 1984.

[29] L. N. Shevrin, About some classes of abstract automata, *Uspehi matem. nauk* **17:6 108** (1962), p. 219 (in Russian).

[30] P. H. Starke, *Abstrakte Automaten*, VEB Deutscher Verlag der Wissenschaften, Berlin, 1969.

[31] M. Steinby, On definite automata and related systems, *Ann. Acad. Sci. Fenn.*, Ser. A, I. Mathematica **444**, 1969.

[32] M. Steinby, Classifying regular languages by their syntactic algebras, in: *Results and trends in theoretical computer science* (Graz, 1994), *Lect. Notes in Comput. Sci.*, 812, Springer, Berlin, 1994, pp. 396–409.

Received January, 1998

Acta Cybernetica **13** (1998) 405-411.

On minimal and maximal clones II

László Szabó *†

Abstract

Two minimal clones which generate all operations, and two maximal clones with trivial intersection are given on $2p$–element sets where $p \geq 5$ is a prime number.

1 Introduction

Let A be a fixed universe with $|A| \geq 2$ and let $\mathbf{O}_A$ denote the set of all finitary operations on A. For $1 \leq i \leq n$ let e_i^n denote the n-ary i-th projection (trivial operation). Further let $\mathbf{J}_A = \{e_i^n | 1 \leq i \leq n < \infty\}$. The operations in $\mathbf{O}_A \setminus \mathbf{J}_A$ are called *nontrivial operations*. By a *clone* we mean a subset of $\mathbf{O}_A$ which is closed under superpositions and contains all projections. The set of clones, ordered by inclusion, forms an algebraic lattice $\mathbf{L}_A$ with least element $\mathbf{J}_A$ and greatest element $\mathbf{O}_A$. For A finite $\mathbf{L}_A$ is an atomic and dually atomic lattice with finitely many atoms and coatoms. The atoms and the coatoms of $\mathbf{L}_A$ are called *minimal clones* and *maximal clones*, respectively.

In [4] we showed that for an at least three element finite set A there are three maximal clones with intersection $\mathbf{J}_A$, and there are three minimal clones with join $\mathbf{O}_A$. If $|A|$ is a prime number then there are two maximal clones and two minimal clones with the above properties. Moreover, we formulated the following two problems:

Problem 1 *Find all natural numbers k for which there exist two maximal clones on a k-element set A such that their intersection is $\mathbf{J}_A$.*

Problem 2 *Find all natural numbers k for which there exist two minimal clones on a k-element set A such that their join is $\mathbf{O}_A$.*

This short note is a modest step to answer these problems. Namely, we give two maximal clones with intersection $\mathbf{J}_A$ and two minimal clones with join $\mathbf{O}_A$ on a $2p$-element set A where p is a prime number with $p \geq 5$.

*Research partially supported by Hungarian National Foundation for Scientific Research grants no. OTKA T022876, OTKA T026243 and Scientific Research grant of the Hungarian Education Ministry no. FKFP 0877/1997.

†Bolyai Institute, H-6720 Szeged, Aradi vértanúk tere 1, Hungary

2 Results

We need some more notions. A ternary operation f on A is a *majority operation* if for all $x, y \in A$ we have $f(x,x,y) = f(x,y,x) = f(y,x,x) = x$; f is a *Mal'cev operation* if $f(x,y,y) = f(y,y,x) = x$ for all $x, y \in A$. An n-ary operation t on A is said to be an i-th *semi-projection* ($n \geq 3$, $1 \leq i \leq n$) if for all $x_1, \ldots, x_n \in A$ we have $t(x_1, \ldots, x_n) = x_i$ whenever at least two elements among $x_1, \ldots, x_n$ are equal.

For a finitary relation ρ on A the set of operations preserving ρ forms a clone, and is denoted by $\operatorname{Pol}\rho$.

Theorem 1 *Let $A = \{0, 1, \ldots, 2p-1\}$ where p is a prime number with $p \geq 5$ and put $C = \{0, 1, p, p+2\}$. Let us define a binary relation ρ and a permutation π on A as follows:*

$$\rho = \{(a,a) : a \in A\} \cup (C \times A) \cup (A \times C)$$

and

$$\pi = (0\ 1\ \ldots\ p-1)(p\ p+1\ \ldots\ 2p-1).$$

Then $\operatorname{Pol}\rho$ *and* $\operatorname{Pol}\pi$ *are maximal clones and* $\operatorname{Pol}\rho \cap \operatorname{Pol}\pi = \operatorname{Pol}\{\rho, \pi\} = \mathbf{J}_A$.

Proof: Taking into consideration the list of maximal clones given by I. G. Rosenberg (see e.g. [3]) we have that $\operatorname{Pol}\rho$ and $\operatorname{Pol}\pi$ are maximal clones. We need the following fact which follows immediately from the definitions of C and π: (*)

For any $x, y \in A$, $x \neq y$, there is a $k \in \{0, \ldots, p-1\}$ such that $x\pi^k \in C$ and $y\pi^k \notin C$. First we establish some properties of the operations in $\operatorname{Pol}\{\rho, \pi\}$. Let

$f \in \operatorname{Pol}\{\rho, \pi\}$ be an arbitrary n-ary operation, $n \geq 1$.

Claim 1 $f(A^n) \supseteq \{0, 1 \ldots, p-1\}$ *or* $f(A^n) \supseteq \{p, p+1 \ldots, 2p-1\}$.

This claim follows immediately from the fact that $f \in \operatorname{Pol}\pi$.

Claim 2 $f(C^n) \subseteq C$.

Let $c_1, \ldots, c_n \in C$. By Claim 1, there are $a_1, \ldots, a_n \in A$ such that

$$f(a_1, \ldots, a_n) \notin C \cup \{f(c_1, \ldots, c_n)\}.$$

Then $(c_1, a_1), \ldots, (c_n, a_n) \in \rho$, and therefore $(f(c_1, \ldots, c_n), f(a_1 \ldots, a_n)) \in \rho$. From this, taking into consideration the definition of ρ, it follows that $f(c_1, \ldots, c_n) \in C$.

Claim 3 *f is an idempotent operation.*

Consider the unary operation $g(x) = f(x, \dots, x)$. If $g(0) = 0$ and $g(p) = p$ then $g(x) = x$ for all $x \in A$ and f is an idempotent operation. Indeed, in this case for $k = 0, \dots, p-1$ we get that

$$g(k) = g(0\pi^k) = g(0)\pi^k = 0\pi^k = k$$

and

$$g(p+k) = g(p\pi^k) = g(p)\pi^k = p\pi^k = p+k.$$

Therefore we have to show that $g(0) = 0$ and $g(p) = p$. By Claim 2,

$$g(0), g(1) \in C = \{0, 1, p, p+2\}.$$

It follows that

$$g(0) = g(1\pi^{-1}) = g(1)\pi^{-1} \in C\pi^{-1} = \{p-1, 0, 2p-1, p+1\}$$

and $g(0) = 0$. Similarly,

$$g(p), g(p+2) \in C = \{0, 1, p, p+2\}$$

implies that

$$g(p) = g((p+2)\pi^{-2}) = g(p+2)\pi^{-2} \in C\pi^{-2} = \{p-2, p-1, 2p-2, p\}$$

and $g(p) = p$, completing the proof of Claim 3.

Claim 4 *If f is binary then $f(x, y) \in \{x, y\}$ for all $x, y \in A$.*

Let f be binary and suppose that $f(a, b) = c \notin \{a, b\}$ for some $a, b \in A$. Then, by (*), $a\pi^k \in C$ and $c\pi^k \notin C$ for some k. Put $u = a\pi^k$, $w = c\pi^k$ and $v = b\pi^k$. Then

$$f(u, v) = f(a\pi^k, b\pi^k) = f(a, b)\pi^k = c\pi^k = w \notin C,$$

and therefore, by Claim 2, we have that $v \notin C$. Now $c \neq b$, $(u, v), (v, v) \in \rho$ imply that $w \neq v$ and $(w, v) = (f(u, v), f(v, v)) \in \rho$ which is not valid.

Claim 5 *If f is binary then the restrictions of f to $\{0, 1, \dots, p-1\}$ and to $\{p, p+1 \dots, 2p-1\}$ are projections.*

By Claim 4, $f(0, 1) \in \{0, 1\}$, and without loss of generality we can suppose that $f(0, 1) = 0$. Then

$$f(p-1, 0) = f(0\pi^{-1}, 1\pi^{-1}) = f(0, 1)\pi^{-1} = 0\pi^{-1} = p-1$$

and

$$f(p-2, p-1) = f(0\pi^{-2}, 1\pi^{-2}) = f(0, 1)\pi^{-2} = 0\pi^{-2} = p-2.$$

Let $i \in \{2, \dots, p-2\}$. From

$$(p-1, 0), (0, i) \in \rho, \ (p-1, i) \notin \rho \ \text{ and } \ f(0, i) \in \{0, i\}$$

it follows that

$$(p-1, f(0,i)) = (f(p-1,0), f(0,i)) \in \rho \text{ and } f(0,i) = 0.$$

Similarly, from

$$(p-2,0), (p-1,p-1) \in \rho, \ (p-2,p-1) \notin \rho \text{ and } f(0,p-1) \in \{0, p-1\}$$

it follows that

$$(p-2, f(0,p-1)) = (f(p-2,p-1), f(0,p-1)) \in \rho \text{ and } f(0,p-1) = 0.$$

Hence for any $x \in \{0,1,\ldots,p-1\}$ we have that $f(0,x) = 0$, which together with the fact that $f \in \operatorname{Pol}\pi$ imply that the restriction of f to $\{0,1,\ldots,p-1\}$ is the first projection. One can show by a very similar argument that the restriction of f to $\{p, p+1, \ldots, 2p-1\}$ is also projection.

Claim 6 *If f is binary then f is a projection.*

Taking into consideration Claim 5, we can suppose without loss of generality that the restriction of f to $\{0,1,\ldots,p-1\}$ is the first projection. First we show that the restriction of f to $\{p, p+1, \ldots, 2p-1\}$ is also the first projection. In deed, if the restriction of f to $\{p, p+1, \ldots, 2p-1\}$ is the second projection then from $(2,p), (0,p+1) \in \rho$ we obtain that $(2,p+1) = (f(2,0), f(p,p+1)) \in \rho$ which is not valid.

If f is not the first projection then for some $a \in \{0,1,\ldots,p-1\}$ and $b \in \{p, p+1, \ldots, 2p-1\}$ we have that $f(a,b) = b$ or $f(b,a) = a$. If $f(a,b) = b$ then choose a positive integer k such that $a\pi^k \in C$ and $v = b\pi^k \notin C$. Put $u = a\pi^k$ and $v = b\pi^k$. Now

$$f(u,v) = f(a\pi^k, b\pi^k) = f(a,b)\pi^k = b\pi^k = v \notin C.$$

Since $(2,u), (0,v) \in \rho$ and $2 \neq v$ (because of $v = b\pi^k \in \{p, p+1, \ldots, 2p-1\}$) it follows that $(2,v) = (f(2,0), f(u,v)) \in \rho$ which is not valid.

If $f(b,a) = a$ then choose a positive integer k such that $a\pi^k \notin C$ and $v = b\pi^k \in C$. Put $u = a\pi^k$ and $v = b\pi^k$. Now

$$f(v,u) = f(b\pi^k, a\pi^k) = f(b,a)\pi^k = a\pi^k = u \notin C.$$

Since $(p+1,v), (p,u) \in \rho$ and $p+1 \neq u$ (because of $u = a\pi^k \in \{0,1,\ldots,p-1\}$) it follows that $(p+1,u) = (f(p+1,p), f(v,u)) \in \rho$ which is not valid. Hence f is the first projection.

Claim 7 *f cannot be a Mal'cev operation.*

Indeed, if f is a Mal'cev operation, then $(2,0), (0,0), (0,3) \in \rho$ implies that $(2,3) = (f(2,0,0), f(0,0,3)) \in \rho$ which is not valid.

Claim 8 *f cannot be a nontrivial semi-projection.*

Let f be a nontrivial n-ary semi-projection ($n \geq 3$). We can suppose that f is a first semi-projection. Observe that $f(c, a_2 \ldots, a_n) \in C$ for any $c \in C$ and $a_2, \ldots, a_n \in A$. Indeed, if $c \in C$ and $a_2, \ldots, a_n \in A$ then for any $a \in A$ we have $(c, a), (a_2, c), \ldots, (a_n, c) \in \rho$ which implies that

$$(f(c, a_2 \ldots, a_n), a) = (f(c, a_2, \ldots, a_n), f(a, c, \ldots, c)) \in \rho$$

and $f(c, a_2 \ldots, a_n) \in C$. Since f is not the first projection $f(a_1, \ldots, a_n) = a \neq a_1$ for some $a_1, \ldots, a_n \in A$. Then, by (*), $a_1\pi^k \in C$ and $a\pi^k \notin C$ for some k. It follows that

$$f(a_1\pi^k, \ldots, a_n\pi^k) = f(a_1, \ldots, a_n)\pi^k = a\pi^k,$$

a contradiction.

Claim 9 *f cannot be a majority operation.*

Let f be a majority operation. First observe that $f(a, b, c) \in C$ if at least two elements among a, b, c belong to C. Indeed, if e.g. $a, b \in C$ then for any $x \in A$ from $(a, x), (b, x), (x, 0) \in \rho$ it follows that

$$(f(a, b, c), x) = (f(a, b, c), f(x, x, 0)) \in \rho$$

which implies that $f(a, b, c) \in C$.

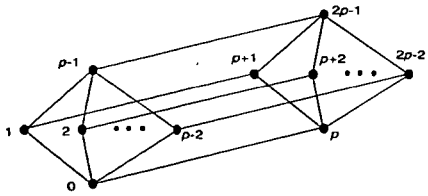
Now let $a, b, c \in A$ be pairwise distinct elements. Clearly, $f(a, b, c)$ is different from at least two of the elements a, b, c, say from a and b. Then, by (*), for some k we have $u = a\pi^k \in C$ and $t = f(a, b, c)\pi^k \notin C$. Put $v = b\pi^k$ and $w = c\pi^k$. Thus

$$f(u, v, w) = f(a\pi^k, b\pi^k, c\pi^k) = f(a, b, c)\pi^k = t$$

and, taking into consideration the above observation, we have that $v \notin C$. Since $f(a, b, c) \neq b$, therefore $v \neq t$ and $(v, t) \notin \rho$. On the other hand $(v, u), (v, v), (0, w) \in \rho$ implies that $(v, t) = (f(v, v, 0), f(u, v, w)) \in \rho$. This contradiction implies that f cannot be a majority operation. Now we are in a position to complete the proof of

the theorem. If $\mathrm{Pol}\{\rho, \pi\} \neq \mathbf{J}_A$ then there is a nontrivial operation in $\mathrm{Pol}\{\rho, \pi\}$ which is either a unary operation or an idempotent binary operation or a majority operation or a Mal'cev operation or a semi-projection (see e.g. [4]). Since , by Claims 3, 6, 7, 8 and 9, these cases cannot occur we have that $\mathrm{Pol}\{\rho, \pi\} = \mathbf{J}_A$. □

Theorem 2 *Let $A = \{0, 1, \ldots, 2p-1\}$ where p is a prime number with $p \geq 5$ and let $(A; \vee, \wedge)$ be the lattice given by the following diagram:*



Let us define a ternary operation d and a permutation π on A as follows:

$$d(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$$

and

$$\pi = (0\ 1\ p+2\ \ldots\ 2p-2\ 2p-1)(p+1\ p\ 2\ \ldots\ p-2\ p-1).$$

Then d and π generate minimal clones such that the clone generated by d and π is $\mathbf{O}_A$.

Proof: Suppose that A, p, d and π satisfy the hypotheses of the theorem. Then it is known that π and d generate minimal clones, respectively (see e.g. [2]). We have to show that $\mathbf{A} = (A; d, \pi)$ is a primal algebra, i.e., every operation on A is a term operation of $\mathbf{A}$.

First observe that $\mathbf{A}$ has no proper subalgebra. Indeed, the proper subalgebras of $(A;\pi)$ are $\{0,1,p+2,\ldots,2p-2,2p-1\}$ and $\{p,p+1,2,\ldots,p-2,p-1\}$ only. Furthermore,

$$d(p+2,p+3,p+4) = p \notin \{0,1,p+2,\ldots,2p-2,2p-1\}$$

and

$$d(2,3,4) = 0 \notin \{p,p+1,2,\ldots,p-2,p-1\}.$$

Since $d(x,y,0) = x \wedge y$ and $d(x,y,2p-1) = x \vee y$ for any $x, y \in A$, therefore the congruence relations of $\mathbf{A}$ and $(A; \vee, \wedge, \pi)$ are the same. One can check easily that $(A; \vee, \wedge)$ has two nontrivial congruence relations only. One of them has two blocks

$$B = \{0,1,\ldots,p-1\} \quad \text{and} \quad C = \{p,p+1,\ldots,2p-1\},$$

and the blocks of the other are

$$\{k, p+k\},\ k = 0,\ldots,p-1.$$

It is easy to check that π does not preserve these two equivalence relations. Hence we have that $\mathbf{A}$ is a simple algebra.

Next we show that the identity map is the only automorphism of $\mathbf{A}$. To show this let τ be an automorphism of $\mathbf{A}$. Since τ is also an automorphism of the algebra $(A; d)$, for any $\Theta \in \mathrm{Con}(A; d) = \mathrm{Con}(A; \vee, \wedge)$ we have that $\Theta\tau \in \mathrm{Con}(A; d)$. It follows that either $B\tau = B$ or $B\tau = C$. Hence $\tau|_B$ is either an automorphism of $(B; d)$ or an i somorphism between $(B; d)$ and $(C; d)$. For any $x \in B\tau$ we have that

$$d(x, 0\tau, (p-1)\tau) = d(x\tau^{-1}, 0, p-1)\tau = (x\tau^{-1})\tau = x.$$

Using this fact it is easy to show that either $\{0\tau, (p-1)\tau\} = \{0, p-1\}$ or $\{0\tau, (p-1)\tau\} = \{p, 2p-1\}$. If $0\tau = p-1$ then

$$1\tau = (0\pi)\tau = 0(\pi\tau) = 0(\tau\pi) = (0\tau)\pi = (p-1)\pi = p+1 \quad \text{and} \quad B\tau \neq B, C.$$

If $0\tau = p$ then

$$1\tau = (0\pi)\tau = 0(\pi\tau) = 0(\tau\pi) = (0\tau)\pi = p\pi = 2 \quad \text{and} \quad B\tau \neq B, C.$$

If $0\tau = 2p-1$ then

$$1\tau = (0\pi)\tau = 0(\pi\tau) = 0(\tau\pi) = (0\tau)\pi = (2p-1)\pi = 0 \quad \text{and} \quad B\tau \neq B, C.$$

Taking into consideration that $B\tau = B$ or $B\tau = C$, it follows that $0\tau = 0$. Since the set of fixed points of τ is a subalgebra of $\mathbf{A}$ therefore τ is the identity map.

No we are in a position to complete the proof. By [5], every finite, simple, surjective algebra without proper subalgebra is either quasiprimal or affine or term equivalent to a matrix power of a unary algebra. Since affine algebras and matrix powers of unary algebras cannot have majority term operations and d is a majority operation, we obtain that $\mathbf{A}$ is quasiprimal (i.e. every operation on A admitting all isomorphisms beetwen subalgebras of $\mathbf{A}$ is a term operation of $\mathbf{A}$). Taking into consideration that $\mathbf{A}$ has no proper subalgebras and nontrivial automorphisms, it follows that $\mathbf{A}$ is a primal algebra. □

References

[1] P. P. Pálfy, L. Szabó and Á. Szendrei, Automorphism groups and functional completeness, *Algebra Universalis* **15** (1982), 385-400.

[2] R. Pöschel and L. A. Kalužnin, *Funktionen- und Relationenalgebren*, VEB Deutscher Verlag d. Wissenschaften, Berlin, 1979.

[3] I. G. Rosenberg, Completeness properties of multiple-valued logic algebras, in: *Computer Science and Multiple-Valued Logic, Theory and Applications* (ed. D. C. Rine), North-Holland (1977); pp. 144-186.

[4] L. Szabó, On minimal and maximal clones, *Acta Cybernetica* **10** (1992), 323–327.

[5] Á. Szendrei, Simple surjective algebras having no proper subalgebras, *J. Austral. Math. Soc.* **48** (1990), 434–454.

Received March, 1998

Acta Cybernetica **13** (1998) 413-421.

Improving Storage Handling of Interval Methods for Global Optimization *

Csallner A. E. †

Abstract

Global nonlinear optimization problems can be solved by interval subdivision methods with guaranteed reliability. These algorithms are based on the branch-and-bound principle and use special storage utilities for the paths not pruned from the search tree yet. In this paper the possibilities for the kinds of applied storage units are discussed.

If no ordering is kept in the storage unit then the dependence of the number of operations demanded by the storage on the iterations completed is quadratic in worst case. On the other hand, ordering the elements as it is necessary for choosing new elements from the storage unit for backtracking, the worst case for the number of storage operations done to the k-th iteration has the magnitude $k \log k$. The hybrid method defined in this paper satisfies the same complexity properties. It is also proved that the $k \log k$ magnitude is optimal.

1 Global Optimization and Interval Methods — Introduction

The global optimization problem can be defined in general as follows:

$$\min_{x \in X} f(x) \tag{1}$$

where X is a — possibly multidimensional — interval. If we denote the set of real intervals by $\mathbb{I}$ and $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function of the problem, then $X \in \mathbb{I}^n$. Note, that a great class of real-life bound-constrained global optimization problems are covered by (1), e.g., problems where the parameters are given with tolerances or if the optimizers are supposed to be inside a parameter region [2].

Problem (1) can be solved with verified accuracy with the aid of interval methods (see, e.g., [1, 5, 6, 7, 8]). These methods are based on the well-known branch-and-bound principle. Thus, a search tree is built where the whole search region — the

*This work has been supported by the Grants OTKA T017241, F025743, and FKFP 0739/1997. Presented at the Conference of PhD Students on Computer Science, July 18-22 1998, Szeged.

†Department of Computer Science, Juhász Gyula Teacher Training College, 6725 Szeged, Boldogasszony sgt. 6, Hungary, e-mail: csallner@jgytf.u-szeged.hu

interval X — is the root and the particular levels consist of subintervals which are partitions of their parents in the tree. Those branches that cannot be pruned have to be stored for later treatment. The kind of the storage method used can be of great importance in performance if the increase of the number of intervals to be stored is considerable.

The following outlined algorithm is a model algorithm for interval subdivision methods for global optimization.

Algorithm 1 Model algorithm for interval subdivision methods for global optimization.

Step 1 Let S be an empty storage unit, the actual box $A := X$, and the iteration counter $k := 1$.

Step 2 Subdivide A into finite number $s \geq 2$ of subintervals A_i satisfying $A = \cup A_i$ so that $\mathrm{int}(\mathrm{A_i}) \cap \mathrm{int}(\mathrm{A_j}) = \emptyset$ for all $i \neq j$ where 'int' denotes the interior of a set.

Step 3 Let $S := S \cup \{A_i\}$.

Step 4 Discard certain elements from S.

Step 5 Choose a new $A \in S$ and delete it from S, $S := S \setminus \{A\}$.

Step 6 While termination criteria do not hold let $k := k + 1$ and go to Step 2.

Steps 3, 4, and 5 are using S, and their running time depends obviously on the storage handling of the algorithm. In the following we shall concentrate on these parts of the algorithm.

2 Worst Cases in Storage Handling of Interval Subdivision Methods

Because the efficiency of a branch-and-bound method depends highly on which branch, i.e., stored element is chosen as next to be treated, two basically different principles can be applied to handle the list. The first is to keep the list ordered and always pick up the first (or last) element in that ordering, while the second is to let the list be unordered and search for the next element in each step a new one is needed. The former saves computational time at picking up the elements, the latter at storing them. The time necessary for storage handling can be calculated in both cases.

We shall assume that Step 4 is not involved in the considered algorithm. In practice, this is the case in most implementations because cleaning up the stored elements is usually done when choosing a new A or before storing the new A_i intervals.

If the stored elements are unordered then Step 3 consumes some $c_1 s$ operations, and Step 5 some $c_2|S|$. If there is an ordering present which provides a constant time operation for finding the new element A in Step 5 then with a most efficient method the newly arisen s subintervals are stored in some $c_3 s \log |S|$ time and the new actual interval A is delivered in constant c_4 time. We summarize these results in the following lemma.

Lemma 1 *The worst case time complexity of storage handling of Algorithm 1 is*

$$c_1 s + c_2|S| \tag{2}$$

if S is unordered, and

$$c_3 s \log |S| + c_4 \tag{3}$$

if S is ordered. The results apply to a single iteration.

The algorithm keeps running for some k_0 iteration steps, and our goal is now to determine the total time consumption of the storage handling for the whole running time of the algorithm.

2.1 Unordered Storage Handling

If the elements are unordered then we can simply consider the storage unit as a sequential list. The next theorem says that in this case the number of storage operations depends quadratically on the iterations completed.

Theorem 2 *If S of Algorithm 1 is unordered and the process of finding a new actual interval A depends on a certain property of the list elements then the time complexity of storage operations up to the k_0-th iteration is*

$$T(s, k_0) = \mathcal{O}(s k_0^2). \tag{4}$$

Proof. From (2) of Lemma 1 it follows that a single iteration step uses $c_1 s + c_2|S|$ operations for appending the newly arisen intervals to the list and to find a new actual interval, where c_1 and c_2 are independent constants. The total number of storage operations done till the k_0-th iteration is hence

$$T(s, k_0) = \sum_{k=1}^{k_0} (c_1 s + c_2|S|). \tag{5}$$

Let us check how Algorithm 1 works. After the first iteration at most s elements are put onto the empty list, and one of them is picked up as the new actual interval in the same iteration. Thus, the number of list elements becomes $s - 1$. The list grows in every iteration by at most $s - 1$. Hence, at the end of some k-th iteration in worst case the number of list elements equals

$$|S| = k(s - 1). \tag{6}$$

Now putting (5) and (6) together we have

$$T(s,k_0)=\sum_{k=1}^{k_0}(c_1 s+c_2 k(s-1))=c_1 s k_0+c_2(s-1)\frac{k_0^2+k_0}{2}, \tag{7}$$

delivering (4) of the assertion. □

Note, that the proof does not presume the way of either subdividing or choosing a new actual box. Hence, all methods covered by the model Algorithm 1 obey Theorem 2.

2.2 Ordered Storage Handling

The following result is only sharp if in worst case more than a number of operations linear in the storage unit's cardinality is needed to keep the elements in the storage unit ordered.

Theorem 3 *If S of Algorithm 1 is kept ordered and the selection of the new actual intervals presume considering the elements' ordering then the time complexity of storage handling up to iteration k_0 is*

$$T(s,k_0)=\mathcal{O}(sk_0(\log s+\log k_0)). \tag{8}$$

Proof. (3) of Lemma 1 says that storage handling costs $c_3 s\log|S|+c_4$ in each iteration, where c_3 and c_4 are independent constants. Hence, summing this to the k_0-th iteration we get

$$T(s,k_0)=\sum_{k=1}^{k_0}(c_3 s\log|S|+c_4). \tag{9}$$

On the other hand, from (6) of the proof of Theorem 2 (9) can be extended to

$$T(s,k_0) = \sum_{k=1}^{k_0}(c_3 s\log(k(s-1))+c_4)= \tag{10}$$

$$= c_3 s\sum_{k=1}^{k_0}\log k+c_3 s k_0\log(s-1)+c_4 k_0. \tag{11}$$

The first term of (11) can be bounded from above using the following inequality:

$$\sum_{k=1}^{k_0}\log k \le \int_1^{k_0+1}\log x\,dx=\frac{1}{\ln a}[x\ln x-x]_1^{k_0+1}= \tag{12}$$

$$= \frac{1}{\ln a}(k_0+1)\ln(k_0+1)-\frac{k_0}{\ln a}, \tag{13}$$

where a denotes the base of the logarithm function in question. Note, that its value cannot influence the magnitude of the formula.

Substituting (13) into (11) provides the magnitudes that had to be proved. □

If s is considered as a constant then till finishing the k_0-th iteration the magnitude of storage handling's time consumption is $k_0 \log k_0$ in worst case, provided that any element can be inserted to the ordered storage unit in logarithmic time.

However, the time consumption's dependence on s is higher by a $\log s$ factor than in the unordered case (see Theorem 2), though this might not mean heavy differences by the usually small values of s.

2.3 Hybrid Storage Handling

A further storage handling method can be used which has not been mentioned here yet. In reality, this is not a new one but a mixture of those from Subsection 2.1 and 2.2. The idea is [4] to keep some p_0 elements from the storage unit ordered. These elements have to be the first ones regarding to the ordering of the whole. Hence, the new actual interval can always be chosen as the first of the ordered part. When inserting newly arisen intervals, one of them can supply for the ordered part. Otherwise a new element for the ordered part can be searched for in the unordered part. In worst case it can occur that for a substantial number of iterations the ordered part can only be refilled from the unordered part consuming $\mathcal{O}(|S| - p_0)$ time in each iteration.

Therefore, let us consider the following modification; we shall fix the value of p_0 and store only the first p $(1 \leq p \leq p_0)$ elements ordered. The remaining $|S| - p$ elements are stored in a simple list.

Thus, every time a new interval is needed, it can be reached in constant time, since the first element of the ordered part does it. The other direction, namely, storing new elements is a little bit more difficult. For each new element it is checked whether it can be inserted into the ordered part of S. If it is the case, the element is inserted. If $p = p_0$ held before the insertion, the last element is moved to the unordered part. If the new element is greater at the present ordering than any from the ordered part — assuming the ordering is increasing — then it is pushed simply onto the unordered part. This procedure can be done in some $c_5 \log p_0$ time in worst case. Since there are at most s new elements to be inserted, the total amount of time needed for the storage operations is

$$c_5 s \log p_0 + c_6 \tag{14}$$

together with the constant number of operations for picking up the new actual interval. If p_0 was considered as an independent constant, (14) could lead to the conclusion that for an arbitrary number k_0 of iterations the time complexity of storage operations is linear in the variable k_0. In worst case, however, the ordered part can become empty forcing the new element to be chosen from the unordered part. Theoretically this can occur arbitrary many times. Thus, we should enhance the performance, namely, by doing the following two things:

1. We compensate the missing elements of the ordered part only if the ordered part runs empty, and then it is filled up with p_0 elements from the unordered part.

2. We do not fix p_0 directly as a constant, but say that it is always a κ proportion of $|S|$, where κ is a constant.

We shall call this algorithm the *hybrid* method. The time complexity of this method is described in Theorem 4.

Theorem 4 *Let us implement Algorithm 1 with the storage handling described above as the hybrid method. Then the time complexity of storage handling operations is*

$$T(s, k_0) = \mathcal{O}(k_0 \log(sk_0)) \tag{15}$$

in worst case.

Proof. Upon the assumptions the elements are stored in two disjunct units, one is ordered, the other is not. If the number of all the elements is $|S|$ then the ordered part consists of at most $p_0 = \kappa|S|$ elements, where $0 < \kappa < 1$ hold.

The procedure of storage handling involves two stages. In worst case, the first stage is done through p_0 iterations, i.e., until the ordered part becomes empty. The next stage consists of a single iteration, where the ordered part is rebuilt.

In the second stage filling up the ordered part is done as follows. First of all, the first p_0 elements are ordered which needs

$$t_1 = c_5 p_0 \log p_0 \tag{16}$$

time. After this, the remaining $|S| - p_0$ elements have to be inserted into the ordered part if possible, each with a $\log p_0$ time complexity. This can be done with the following time consumption:

$$t_2 = c_6(|S| - p_0) \log p_0. \tag{17}$$

From (6) of Theorem 2 we know that in worst case the number of list elements after the k_0-th iteration is $|S| = k_0(s-1)$. Let us apply this to (17) and add it to (16):

$$t = c_5 p_0 \log p_0 + c_6(k_0(s-1) - p_0) \log p_0. \tag{18}$$

It is known that at the iteration in question p_0 equals $\kappa|S|$ with a given κ. But citing (6) of the proof of Theorem 2 again we have

$$p_0 = \kappa|S| = \kappa k_0(s-1) \tag{19}$$

Let us use this to (18):

$$t = (c_5\kappa + c_6(1-\kappa))k_0(s-1)\log(\kappa k_0(s-1)) \tag{20}$$

These t operations only have to be done periodically after each p_0 iterations, where p_0 grows monotonously. Applying amortized analysis t can be apportioned among the next p_0 iterations to get the time complexity of storage handling for a single iteration in average. If for this the actual $|S|$ is used, we get an underestimation due to the increasing value of $|S|$ and hence p_0. Thus, for the time complexity the following holds:

$$k_0^{-1}T(s,k_0) \leq \frac{t}{p_0} + c_7 = \frac{t}{\kappa k_0(s-1)} + c_7, \tag{21}$$

where c_7 is the time consumption of the first stage. Applying this to the form for t calculated in (20) we have

$$k_0^{-1}T(s,k_0) \leq \frac{c_5\kappa + c_6(1-\kappa)}{\kappa}\log(\kappa k_0(s-1)) + c_7, \tag{22}$$

delivering the magnitude of (15) after k_0 iterations. □

The result of Theorem 4 is a nice enhancement in storage handling. To reach it we had to make two further assumptions previous to the theorem. None of them can be left away unless damaging the bound of (15). Namely, if the ordered part is supplied continually then in worst case the time complexity becomes the same as in the unordered case (see Theorem 2). On the other hand, if p_0 is a value independent from $|S|$ then the amortized analysis leaves a term containing k_0 on first degree in the formula also resulting in a quadratic time complexity in k_0 similar to (4).

From the three investigated methods the dependence on s is far the best with its $\log s$ complexity. This magnitude means that in practice — where $s > 8$ hardly ever occurs — the influence of s is unimportant.

Moreover, for the dependence on the number k_0 of iterations the following theorem holds.

Theorem 5 *For the time complexity dependence on the number k_0 of iterations the magnitude $k_0 \log k_0$ is optimal in worst case.*

Proof. It will be proved that if it was not optimal then an algorithm could be given to order n elements in less than $\mathcal{O}(n \log n)$ time in worst case.

Let the n data to be ordered be denoted by $a_1, a_2, \ldots, a_n$. We can determine the $max := \max_{i=1,\ldots,n} a_i$ value in linear time, and let a be greater regarding to the ordering than max. Then the algorithm is the following. We take a list handling method for interval subdivision methods which has a smaller time complexity than $k_0 \log k_0$ provided the algorithm is stopped after k_0 iterations. Now we place the data into the storage unit and begin to select elements from it as the ordering desires it. Instead of each element removed at least two further instances of the value a is put into the storage unit. After n iterations the original data are taken from the storage unit in the right order in less than $n \log n$ time which is a contradiction. □

Corollary 6 *The ordered and hybrid list handling methods for interval branch-and-bound algorithms are both optimal.*

2.4 List Handling of Hansen Methods

In the previous three subsections it has been assumed that the sequence of storing and recalling elements of the storage unit have not necessarily the same order. However, E.R. Hansen's subdivision method [5, 8] selects the oldest element waiting in the storage unit — which is a simple FIFO list in this case — in Step 5 of Algorithm 1, thus, the time complexity for a single iteration is constant. The time complexity after k_0 iterations is linear in the variable k_0. The reason for using the algorithm of Hansen yet quite rarely is that its convergence speed is in best case the same (see [3]) as of all efficiently converging interval subdivision methods in worst case.

3 Best Cases and Concluding Remarks

It is obvious that no storage handling can perform better than linearly depending on the number of iterations since there must be a few storage handling acts in every iteration. A straight consequence of this fact and of the properties discussed in Subsection 2.4 is that the Hansen methods achieve this linear dependence.

Upon the worst cases the unordered and mixed storage handling methods are the worst with their quadratic dependence. For the unordered handling this is the best case, as well, because looking up all list elements for choosing the new actual box cannot be avoided, at all. For the mixed handling it can occur that a newly arisen subinterval can always immediately be inserted into the ordered part consuming constant time in k_0. Thus, the best case behavior shows linear dependence.

The same thoughts lead to the statements about the ordered and hybrid handling methods. These methods both have linear time complexity in best case concerning the number of iterations made.

The optimal storage handling time complexity is $k_0 \log k_0$ — where k_0 denotes the iteration of termination — for interval branch-and-bound methods if the order of the sequence of resulting elements is not necessarily the same as that of the recalled elements.

This optimal complexity is obtained for the ordered storage types and the hybrid method.

If the algorithm itself provides the ordering of new elements then the worst case is the same as the best case, i.e., linear regarding to the iterations completed (Hansen algorithm).

The best and worst cases for the rest of the methods are summarized below:

Method	Best Case	Worst Case
Unordered	k_0^2	k_0^2
Ordered	k_0	$k_0 \log k_0$
Mixed	k_0	k_0^2
Hybrid	k_0	$k_0 \log k_0$

Note, that the worst case of the ordered storage handling is only true if a best implementation, i.e., a balanced search tree is used. Otherwise the worst case can also grow to k_0^2 as for the unordered case.

Average cases cannot be treated simply deriving them from the best and worst cases, respectively, since the interval branch-and-bound methods can be applied to every programmable function and thus their influence to the storage's behavior cannot be predicted in general. Numerical tests are in progress but they are not at hand at the present.

References

[1] Alefeld G. and Herzberger J. (1983), *Introduction to Interval Computations*, Academic Press, New York.

[2] Csallner A.E. (1993), *Global Optimization in Separation Network Synthesis*, Hungarian Journal of Industrial Chemistry, 21, pp. 303–308.

[3] Csallner A.E. and Csendes T. (1995), *Convergence Speed of Interval Methods for Global Optimization and the Joint Effects of Algorithmic Modifications*, IMACS/ GAMM SCAN-95 Conference, p. 33, Wuppertal, Germany, September 26-29, 1995.

[4] Csendes T. and Pintér J. (1993), *The Impact of Accelerating Tools on the Interval Subdivision Algorithm for Global Optimization*, European Journal on Operational Research, 65, pp. 314–320.

[5] Hansen E.R. (1992), *Global Optimization Using Interval Analysis*, Marcel Dekker, New York.

[6] Kearfott R.B. (1996), *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht.

[7] Moore, R.E. (1966), *Interval Analysis*, Prentice Hall, Englewood Cliffs NJ.

[8] Ratschek H. and Rokne J. (1993), *Interval Methods*, In: Handbook of Global Optimization, Horst R. and Pardalos P.M. (eds.), Kluwer, Dordrecht, pp. 751–828.

Acta Cybernetica **13** (1998) 423-429.

Minimizing the number of tardy jobs on a single machine with batch setup times *

Günter Rote † Gerhard J. Woeginger‡

Abstract

This paper investigates a single-machine sequencing problem where the jobs are divided into families, and where a setup time is incurred whenever there is a switch from a job in one family to a job in another family. This setup only depends on the family of the job next to come and hence is sequence independent. The jobs are due-dated, and the objective is to find a sequence of jobs that minimizes the number of tardy jobs.

The special case of this problem where in every family the jobs have at most two different due dates is known to be $\mathcal{NP}$-complete [Bruno & Downey, 1978]. The main result of this paper is a polynomial time algorithm for the remaining open case where in every family all the jobs have the same due date. This case may be formulated as a dual resource allocation problem with a tree-structured constraint system, which can be solved to optimality in polynomial time.

Keywords. sequencing; scheduling; batch setup times; number of tardy jobs.

1 Introduction

This paper deals with the following scheduling problem. There are n jobs $J_1, \ldots, J_n$ that are to be processed without interruption on a single machine. All jobs are available for processing at time zero. The set of jobs is divided into F families; a setup time s_f is associated to each family $f = 1, \ldots, F$. Whenever a job in family f is processed, this incurs the setup time s_f unless another job from the same family is processed immediately before this job. The machine can execute at most one job at a time, and it cannot perform any processing while undergoing a setup. Job J_j $(j = 1, \ldots, n)$ has a positive integer processing time p_j, and an integer due date d_j. In a schedule σ, we denote by $C_j(\sigma)$ the completion time of job J_j $(j = 1, \ldots, n)$.

*This research has been supported by the START program Y43-MAT of the Austrian Ministry of Science, and by the Spezialforschungsbereich *Optimierung und Kontrolle*, Projektbereich *Disckrete Optimierung*.

†Institut für Mathematik B, TU Graz, Steyrergasse 30, A-8010 Graz, Austria, e-mail: rote@opt.math.tu-graz.ac.at.

‡Institut für Mathematik B, TU Graz, Steyrergasse 30, A-8010 Graz, Austria, e-mail: gwoegi@opt.math.tu-graz.ac.at.

If $C_j(\sigma) > d_j$, then job J_j is *tardy* and we set $U_j = 1$. If $C_j(\sigma) \leq d_j$, then job J_j is processed *on-time* and we set $U_j = 0$. The objective is to find a processing order of the jobs that minimizes $\sum_{j=1}^{n} U_j$, i.e. the number of tardy jobs. In the standard scheduling notation (cf. Lawler, Lenstra, Rinnooy Kan & Shmoys [5] and Potts & van Wassenhove [8]), this problem is denoted by $1 \mid s_f \mid \sum U_j$. For related problems and for practical applications involving batch setup times, the interested reader is referred to Monma & Potts [6], Potts & van Wassenhove [8], and Webster & Baker [9].

A special case of $1 \mid s_f \mid \sum U_j$ is the feasibility testing problem, i.e. the problem of deciding whether there is a feasible schedule in which *all* jobs of a given instance are on-time. Bruno & Downey [1] prove that the feasibility testing problem is $\mathcal{NP}$-hard, even if there are only two distinct deadlines per family. An instance of $1 \mid s_f \mid \sum U_j$ where in every family all jobs have the same due date, is said to have *uniform family due dates*. In this paper we will show that the problem with uniform family due dates is solvable in polynomial time. This special case is sufficiently general to contain the problem $1 \mid\mid \sum U_j$ *without* batch setup times (in $1 \mid\mid \sum U_j$, every jobs forms its own family and all family setup times are zero). Hence, our result generalizes the well-known polynomial time algorithm of Moore [7].

Our solution approach to $1 \mid s_f \mid \sum U_j$ is as follows: We formulate $1 \mid s_f \mid \sum U_j$ with uniform family due dates as a dual resource allocation problem with tree-structured constraints (cf. Section 3). Since this dual resource allocation problem can be solved in polynomial time by dynamic programming (cf. Section 2), the scheduling problem itself can be solved in polynomial time.

2 A dual resource allocation problem

The resource allocation problem (cf. Ibaraki & Katoh [3]) is a well-known optimization problem with a (possibly) complex objective function under a single, extremely simple constraint. In the dual resource allocation problem (cf. Katoh, Ibaraki & Mine [4] or Section 10.1 in [3]), the roles are exchanged and the objective function is simple whereas the constraint system may be messy. In this section, we investigate the following dual resource allocation problem (DAP).

$$\text{(DAP)} \qquad \begin{array}{ll} \max & \sum_{f=1}^{F} x_f \\ \text{s.t.} & \begin{cases} \sum_{f \in S} g_f(x_f) \leq c_S & \text{for all } S \in \mathcal{S} \\ 0 \leq x_f \leq n_f & f = 1, \ldots, F \\ x_f \text{ integer} & f = 1, \ldots, F \end{cases} \end{array}$$

For $1 \leq f \leq F$, the function $g_f : [0, n_f] \to \mathbb{R}$ is an arbitrary function which is specified as an ordered list of pairs $(x, g_f(x))$, $x = 0, \ldots, n_f$. The values n_f, $1 \leq f \leq F$, are positive integers. The set system $\mathcal{S}$ is a system of non-empty sets over $\{1, \ldots, F\}$. For every $S \in \mathcal{S}$, the value c_S is an arbitrary real number.

Moreover, denote $n = \sum_{f=1}^{F} n_f$. Observe that $n \geq F$ holds and that by the specification of the functions g_f, the numbers n_f and n are essentially encoded in unary.

Proposition 2.1 *The dual allocation problem (DAP) is an $\mathcal{NP}$-hard problem.*

Proof. The statement may be proved e.g. via a reduction from INDEPENDENT SET IN GRAPHS (cf. Garey & Johnson [2]): Given a graph $G = (V, E)$, find the maximum number of pairwise non-adjacent vertices. For every vertex $v_f \in V$, introduce a corresponding variable x_f in (DAP) with the interpretation "$x_f = 1$" if v_f belongs to the independent set and "$x_f = 0$" otherwise. Moreover, set $n_f = 1$, $g_f(0) = 0$ and $g_f(1) = 1$. For every edge $e = (v_f, v_h)$ introduce the set $\{f, h\}$ in $\mathcal{S}$ and set $c_{\{f,h\}} = 1$. Then the optimal objective value of (DAP) yields the size of the maximum independent set in G. □

A set system $\mathcal{S}$ is called *tree-structured* if $\emptyset \notin S$ and for all $S', S'' \in \mathcal{S}$,

$$S' \subseteq S'' \quad \text{or} \quad S'' \subseteq S' \quad \text{or} \quad S' \cap S'' = \emptyset.$$

With a tree-structured set system $\mathcal{S}$, we associate a directed in-forest $\mathcal{F}(\mathcal{S})$ as follows: For every set $S \in \mathcal{S}$ the forest contains a corresponding vertex; in the following we will not distinguish between a set S and its corresponding vertex. There is a directed edge from a set S' to another set S'' in $\mathcal{F}(\mathcal{S})$ if and only if $S' \subset S''$ and there is no S''' in $\mathcal{S}$ with $S' \neq S''' \neq S'$ and $S' \subset S''' \subset S''$. Clearly, every vertex in $\mathcal{F}(\mathcal{S})$ has out-degree at most one. Adding all singleton sets to $\mathcal{S}$ does not destroy the tree-structured property. But then, the forest $\mathcal{F}(\mathcal{S})$ has F leaves, and the remaining vertices have indegree at least 2. It follows that a tree-structured family $\mathcal{S}$ contains at most $2F - 1$ sets.

Lemma 2.2 *For any instance $I = (n_f, g_f, \mathcal{S}, c_S)$ of (DAP) with tree-structured $\mathcal{S}$, one can construct in $O(n + F^2)$ time another instance $I' = (n_f, g_f, \mathcal{S}', c'_S)$ of (DAP) such that the following conditions are fulfilled.*

(C1) I and I' are equivalent, i.e. they have the same set of optimal solutions and the same optimal objective value.

(C2) The set system $\mathcal{S}'$ in I' is tree-structured; $|\mathcal{S}'| = 2F - 1$ holds; the in-forest $\mathcal{F}(\mathcal{S}')$ associated with $\mathcal{S}'$ is a binary in-tree.

Proof. We construct I' in two steps by adding more sets to $\mathcal{S}$. We set the right-hand sides c'_S of all the corresponding new inequalities to the global upper bound $c^* = \sum_{f=1}^{F} \max\{0, \max_{0 \leq x \leq n_f} g_f(x)\}$, which makes them redundant. Initialize $\mathcal{S}' := \mathcal{S}$ and for all $S \in \mathcal{S}$ set $c'_S = c_S$. If $\{f\} \notin \mathcal{S}'$ for some $f \in \{1, \ldots, F\}$ then add the new set $\{f\}$ to $\mathcal{S}'$. If $\{1, \ldots, F\} \notin \mathcal{S}'$ then add the new set $\{1, \ldots, F\}$ to $\mathcal{S}'$.

In the second step, repeat the following procedure as long as some vertex S in $\mathcal{F}(\mathcal{S}')$ has three or more in-going edges: Let S_{i_1} and S_{i_2} be two arbitrary children of

S in $\mathcal{F}(\mathcal{S}')$; add the set $S' = S_{i_1} \cup S_{i_2}$ to $\mathcal{S}'$. By iterating this procedure, eventually every interior vertex in $\mathcal{F}(\mathcal{S}')$ will have in-degree two and condition (C2) will be fulfilled. This completes the construction of instance I'. It can be verified that I' is equivalent to I and hence, conditions (C1) and (C2) are both fulfilled.

It remains to discuss the time complexity. The first step is easily done in $O(n + F)$ time. In the beginning of the second step, compute the current forest $\mathcal{F}(\mathcal{S}')$ as follows. First construct a simple, undirected, loopless auxiliary graph with vertex set $\mathcal{S}'$: For every $f = 1, \ldots, F$ and for every $S', S'' \in \mathcal{S}'$ with $f \in S'$ and $f \in S''$, put an edge between S' and S'' into the auxiliary graph. The auxiliary graph can be constructed in $O(F^2)$ overall time. Then for $S \in \mathcal{S}'$, $S \neq \{1, \ldots, F\}$, the unique out-going edge in $\mathcal{F}(\mathcal{S}')$ goes to the set S' where (i) S' is adjacent to S in the auxiliary graph, (ii) $|S'| > |S|$, and (iii) $|S'|$ is smallest possible under these conditions. In this way, the forest $\mathcal{F}(\mathcal{S}')$ for $\mathcal{S}'$ at the beginning of the second step can be computed in $O(F^2)$ time from the auxiliary graph. Getting rid of the vertices with in-degree greater than two can be done by locally manipulating $\mathcal{F}(\mathcal{S}')$; it is routine to implement it in $O(F^2)$ overall time. □

Theorem 2.3 *The special case of the dual allocation problem (DAP) where $\mathcal{S}$ is tree-structured is solvable in $O(n^2)$ time.*

Proof. First we apply Lemma 2.2 to get in $O(n + F^2)$ time an equivalent instance where $\mathcal{F}(\mathcal{S})$ is a binary tree. Let $S_1, \ldots, S_{2F-1}$ be an enumeration of the sets in $\mathcal{S}$, such that $S_i \subset S_j$ implies $i \leq j$. For $S \in \mathcal{S}$, let $n(S) = \sum_{f \in S} n_f$.

The remaining argument will be done by dynamic programming. Define a two-dimensional array $A[i, \ell]$ where $1 \leq i \leq 2F - 1$ and $0 \leq \ell \leq n$ with the following meaning: The value $A[i, \ell]$ is the smallest g^* for which there exist values x_f^*, $f \in S_i$, such that

(A1) $\sum_{f \in S} g_f(x_f^*) \leq c_S$ holds for all $S \in \mathcal{S}$, $S \subseteq S_i$.
(A2) $\sum_{f \in S_i} x_f^* = \ell$.
(A3) $\sum_{f \in S_i} g_f(x_f^*) = g^*$.

If no values x_f^* fulfilling (A1) and (A2) exist, then $A[i, \ell] = +\infty$. This happens for example when $\ell > n(S_i)$. Hence from now on, we will only deal with entries $A[i, \ell]$ for which $\ell \leq n(S_i)$. We compute the entries $A[i, \ell]$ in increasing order of i. If $|S_i| = 1$, let $S_i = \{f\}$ and set for $0 \leq \ell \leq n_f$

$$A[i, \ell] = \begin{cases} g_f(\ell) & \text{if } g_f(\ell) \leq c_{S_i} \\ +\infty & \text{otherwise.} \end{cases} \tag{1}$$

If $|S_i| > 1$, let $S_i = S_a \cup S_b$ with $a < b < i$, where S_a and S_b are the two children of S_i in $\mathcal{F}(\mathcal{S})$. Then for $\ell \leq n(S_i)$,

$$A[i, \ell] = \begin{cases} \min \{ A[a, k] + A[b, \ell - k] : 0 \leq k \leq n(S_a),\ 0 \leq \ell - k \leq n(S_b) \} \\ +\infty \qquad \text{if this minimum is greater than } c_{S_i}. \end{cases} \tag{2}$$

It can be verified that with the above definitions, (A1)–(A3) are always fulfilled for $g^* = A[i, \ell]$. In the end, the optimal objective value of (DAP) equals the maximum ℓ for which $A[2F-1, \ell]$ takes a finite value.

Let us analyze the time needed to compute all values $A[i, \ell]$. Denote by $T(i)$ the total time needed to handle all finite entries $A[j, \ell]$ with $0 \leq \ell \leq n$ and $S_j \subseteq S_i$. Then for $|S_i| = 1$ with $S_i = \{f\}$, (1) implies that

$$T(i) \;=\; \text{const}_1 \cdot n_f \;=\; \text{const}_1 \cdot n(S_i). \tag{3}$$

If $|S_i| > 1$, let S_a and S_b be the two children of S_i in $\mathcal{F}(\mathcal{S})$. Note that $a < b < i$, that $S_i = S_a \cup S_b$, and that $n(S_i) = n(S_a) + n(S_b)$ holds. We claim that

$$T(i) \;=\; T(a) + T(b) + \text{const}_2 \cdot n(S_a) \cdot n(S_b). \tag{4}$$

This can be seen as follows. The time $T(i)$ consists of the total time for handling all entries $A[j, \ell]$ with $0 \leq \ell \leq n$ and $S_j \subseteq S_a$ or $S_j \subseteq S_b$, plus the total time for handling all entries $A[i, \ell]$ with $0 \leq \ell \leq n$. For every α, $0 \leq \alpha \leq n(S_a)$, and for every β, $0 \leq \beta \leq n(S_b)$, in (2) there is exactly one step performed with $k = \alpha$ and $\ell - k = \beta$. Hence, the total time for handling the entries $A[i, \ell]$ with $0 \leq \ell \leq n(S_i)$ is proportional to $n(S_a) \cdot n(S_b)$. Hence, (4) indeed holds. By induction, one proves from (3) and (4) that

$$T(i) \;\leq\; \text{const} \cdot n(S_i)^2.$$

Consequently, the total time $T(2F-1)$ needed for computing all entries is $O(n^2)$. Since $F \leq n$, the time spent on applying Lemma 2.2 is also $O(n^2)$. Summarizing, this yields the running time claimed in the statement of the theorem.

Finally, we remark that by storing appropriate auxiliary information in the dynamic program and by doing some backtracking, one can also explicitly compute the values x_f in an optimal solution; this increases the running time by only a constant factor. Since these are standard techniques, we do not elaborate on them. □

3 Solution of the scheduling problem

In this section we discuss the scheduling problem $1 \mid s_f \mid \sum U_j$ that has been defined in the introduction. The following observation follows via straightforward job interchange arguments.

Observation 3.1 *For any instance of* $1 \mid s_f \mid \sum U_j$ *with uniform family due dates, there is an optimal schedule of the following form.*

(i) For every family, the on-time jobs of that family are processed consecutively; hence, the setup for each family is performed at most once.

(ii) In each family, the on-time jobs are the shortest jobs of the family. □

For $f = 1, \ldots, F$, denote by d_f the due date of the jobs in family f. Without loss of generality assume that $d_1 \leq d_2 \leq \cdots \leq d_F$. Let n_f, $f = 1, \ldots, F$, denote the number of jobs in family f, and let $p_{f,1} \leq p_{f,2} \leq \cdots \leq p_{f,n_f}$ denote their processing times. Moreover, define

$$g_f(x) = \begin{cases} 0 & \text{if } x = 0 \\ s_f + \sum_{i=1}^{x} p_{f,i} & \text{if } 1 \leq x \leq n_f. \end{cases}$$

For $f = 1, \ldots, F$, introduce $S_f = \{1, \ldots, f\}$ and set $c_{S_f} = d_f$. Define $\mathcal{S} = \{S_1, \ldots, S_F\}$. Finally, denote by x_f the number of on-time jobs from family f, $f = 1, \ldots, F$.

With this choice of parameters, the dual allocation problem (DAP) is equivalent to $1 \,|\, s_f \,|\, \sum U_j$ with uniform family due dates. Moreover, $\mathcal{S}$ is tree-structured and hence Theorem 2.3 implies the main result of this paper:

Theorem 3.2 *The special case of* $1 \,|\, s_f \,|\, \sum U_j$ *with uniform family due dates is solvable in* $O(n^2)$ *time.* □

Acknowledgement. The authors thank Bettina Klinz for several valuable discussions and for providing pointers to the literature on allocation problems.

References

[1] J. Bruno and P. Downey, Complexity of task sequencing with deadlines, set-up times, and changeover costs, *SIAM Journal on Computing* **7**, 1978, 393–404.

[2] M.R. Garey and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[3] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches*, MIT Press, Boston, 1988.

[4] N. Katoh, T. Ibaraki, and H. Mine, Algorithms for a variant of the resource allocation problem, *Journal of the Operations Research Society of Japan* **22**, 1979, 287–299.

[5] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, Sequencing and scheduling: Algorithms and complexity, in: S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin (eds.) *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science 4, North-Holland, Amsterdam, 1993, 445–522.

[6] C.L. Monma and C.N. Potts, On the complexity of scheduling with batch set-up times, *Operations Research* **37**, 1989, 798–804.

[7] J. MOORE, An n job, one machine sequencing algorithm for minimizing the number of late jobs, *Management Science* **15**, 1968, 102–109.

[8] C.N. POTTS AND L.N. VAN WASSENHOVE, Integrating scheduling with batching and lotsizing: a review of algorithms and complexity, *Journal of the Operational Research Society* **43**, 1992, 395–406.

[9] S. WEBSTER AND K.R. BAKER, Scheduling groups of jobs on a single machine, *Operations Research* **43**, 1995, 692–703.

Received January, 1998

Acta Cybernetica **13** (1998) 431-438.

Generalized Dependencies in Relational Databases *

Attila Sali Sr. † Attila Sali ‡§

Abstract

A new type of dependencies in a relational database model introduced in [5] is investigated. If b is an attribute, A is a set of attributes then it is said that b (p,q)-depends on A, in notation $A \stackrel{(p,q)}{\longrightarrow} b$, in a database relation r if there are no $q+1$ tuples in r such that they have at most p different values in each column of A, but $q+1$ different values in b. $(1,1)$-dependency is the classical functional dependency. Let $\mathcal{J}(A)$ denote the set $\{b: A \stackrel{(p,q)}{\longrightarrow} b\}$. The set function $\mathcal{J}: 2^\Omega \longrightarrow 2^\Omega$ becomes a closure if $p=q$. Results on representability of closures by (p,p)-dependencies are presented.

Keywords: relational database, closure, functional dependency, branching dependency, balanced graph

1 Introduction

A relational database system of the scheme $R(A_1, A_2, \ldots, A_n)$ can be considered as a matrix, where the columns correspond to the *attributes* A_i (for example name, date of birth, place of birth etc.), while the rows are the n-tuples of the relation r. That is, a row contains the data of a given *individual*. Let Ω denote the set of attributes (the set of the columns of the matrix). Let $A \subseteq \Omega$ and $b \in \Omega$. We say that b *(functionally) depends* on A (see [1, 2]) if the data in the columns of A determine the data of b, that is there exist no two rows which agree in A but are different in b. We denote this by $A \longrightarrow b$.

Functional dependencies have turned out to be very useful. In the present paper we investigate a more general (weaker) dependency, than the functional dependency, which was introduced in [5].

The general concept to be studied is the (p,q)-dependency of [5] with $p=q$.

*AMS Subject classification Primary 68P15 68R05 Secondary 05D05

†Informatics Centre of Semmelweis University of Medicine Budapest, Kálvária tér 5. H-1089 HUNGARY

‡The work of the second author was supported by the Hungarian National Foundation for Scientific Research grant numbers T016389, and 4267, and European Communities (Cooperation in Science and Technology with Central and Eastern European Countries) contract number CIPACT930113

§Corresponding author. e-mail:sali@math-inst.hu Mathematical Institute of HAS, Budapest P.O.B. 127 H-1364 HUNGARY

Definition 1.1 *Let a relational database system of the scheme $R(A_1, A_2, \ldots, A_n)$ be given. Let $A \subseteq \Omega$ and $b \in \Omega$. We say that b* (p,q)-depends *on A if there are no $q+1$ rows (n-tuples) of r such that they contain at most p different values in each column (attribute) of A, but $q+1$ different values in b.*

For a given relation r (or its matrix M) we define a function from the family of subsets of Ω into itself, as follows.

Definition 1.2 *Let M be the matrix of the given relation r. Let us suppose, that $1 \leq p \leq q$. Then the mapping $\mathcal{J}_{Mpq}: 2^\Omega \to 2^\Omega$ is defined by*

$$\mathcal{J}_{Mpq}(A) = \left\{ b: A \overset{(p,q)}{\longrightarrow} b \right\}.$$

We collect two important properties of the mapping $\mathcal{J}_{Mpq}$ in the following proposition, see [5].

Proposition 1.3 *Let r, Ω, M, p and q as in Definition 1.2. Furthermore, let $A, B \subseteq \Omega$. Then*

$$\begin{array}{ll} (i) & A \subseteq \mathcal{J}_{Mpq}(A) \\ (ii) & A \subseteq B \Longrightarrow \mathcal{J}_{Mpq}(A) \subseteq \mathcal{J}_{Mpq}(B). \end{array}$$

Definition 1.4 *Set functions satisfying (i) and (ii) are called* increasing-monotone functions. *We say that such an increasing-monotone function $\mathcal{N}$ is* (p,q)-representable *if there exists a matrix M such that $\mathcal{N} = \mathcal{J}_{Mpq}$.*

It was also observed in [5] that in the case $p = q$ the set function $\mathcal{J}_{Mpq}$ satisfies a third property

$$(iii) \quad \mathcal{J}_{Mpq}(\mathcal{J}_{Mpq}(A)) = \mathcal{J}_{Mpq}(A) \quad \text{for all} \quad A \subseteq \Omega.$$

Set functions satisfying (i) – (iii) are called *closures* and are widely investigated. In [6] the minimum representation of closures and increasing-monotone functions were investigated. In [7] the connection of the minimum representation and design theoretical constructions was described. Also many open problems were posed.

In the present paper the representability of closures is investigated. In [1] it was proved that functional dependencies and closures are equivalent. However, in [5] it was pointed out, that this no longer holds for general (p,p)-dependencies. It is natural to ask, which closures arise in connection with these weaker dependencies, or putting the question in another way, given a closure $\mathcal{L}$, what are those p's, for which $\mathcal{L}$ is (p,p)- representable. This motivates the following definition. Because only (p,p)-dependencies and (p,p)-representations are considered, in what follows p-dependency and p-representation are written, for the sake of simplicity.

Definition 1.5 *Let $\mathcal{L}$ be a closure on the set Ω. The spectrum* $\mathrm{SP}(\mathcal{L})$ *of $\mathcal{L}$ is defined as follows.*

$$q \in \mathrm{SP}(\mathcal{L}) \iff \mathcal{L} \text{ is } q - \text{representable}$$

Note that $\mathrm{SP}(\mathcal{L}) \subseteq \mathbf{N}$.

The following special type of closure plays an important role in the theory.

Definition 1.6 *Let C_n^k denote the following closure on Ω ($|\Omega| = n$):*

$$C_n^k(X) = \begin{cases} X & \text{if } |X| < k \\ \Omega & \text{otherwise} \end{cases}$$

The following theorem was proved in [5]

Theorem 1.7

1. $\{1,2\} \subseteq \mathrm{SP}(\mathcal{L})$ *for any closure* $\mathcal{L}$.
2. $\mathrm{SP}(C_n^2) = \{1,2\}$ *if* $n > 6$.
3. *If* $|\Omega| = n$ *and* $2n-3 \leq N \in \mathrm{SP}(\mathcal{L})$, *then* $\forall q \geq N$ $q \in \mathrm{SP}(\mathcal{L})$

The purpose of the present paper is to extend Theorem 1.7. The extension yields some quite surprising results about the spectra of closures. The interested reader is referred to [3, 4, 6, 7] for further investigations and open problems.

2 Spectra of closures

It was shown in [5] that for a matrix M $b \in \mathcal{J}_{Mpq}(A)$ implies $b \in \mathcal{J}_{Mp-1q-1}(A)$ provided the matrix has at least $q+1$ distinct entries in each of its columns. This may lead to the expectation that the spectrum of a closure is an interval of the integers. In this section we show the quite surprising fact that the spectrum of a closure may contain an arbitrary number of "holes", i.e., it may be far from being an interval.

Let the $m \times n$ matrix M p-represent the closure $\mathcal{L}$ on Ω. A mapping w from the edges of the complete graph K_m to the subsets of Ω can be defined, as follows. The vertices of K_m are identified with the set of rows of M. For an edge $e = \{i,j\}$ of K_m, let $w(e)$ be the set of positions where rows i and j agree. If $A \subset \Omega$ and $b \in \Omega$ such that $b \notin \mathcal{L}(A)$, then there exist $p+1$ rows $r_1, r_2, \ldots, r_{p+1}$ that contain at most p distinct values in columns of A but they are all different in column b. Equivalently, $b \notin \bigcup_{1 \leq i < j \leq p+1} w(\{r_i, r_j\}) \supset A$. The next lemma, which is an equivalent formulation of Theorem 2.12 of [5] is explained by the above observation.

Lemma 2.1 *Let $\mathcal{L}$ be a closure on Ω. $\mathcal{L}$ is p-representable if and only if there exists a mapping $w: E(K_m) \longrightarrow 2^\Omega$ of the edges of K_m for some m (where $w(e)$ is called the* weight *of edge e) that satisfies the following two properties:*

1. *For any three edges e_1, e_2, e_3 forming a triangle, $w(e_i) \cap w(e_j) \subseteq w(e_k)$ holds for any permutation (i,j,k) of $(1,2,3)$.*
2. *For any $p+1$ vertices of K_m, the union weights of edges spanned by these vertices is closed by $\mathcal{L}$, and every closed set of $\mathcal{L}$ can be obtained as intersections of sets of this type.*

Condition 1. is the necessary and sufficient condition for the existence of a matrix with prescribed edge weights, while condition 2. is that of the p-representation.

First some constructions are presented that show that certain values of p are in $\mathrm{SP}(C_n^k)$. Then we show, that these are all the elements of $\mathrm{SP}(C_n^k)$ provided n is large enough with respect to k. In what follows, edges of K_m of empty weight will be omitted for the sake of simplicity, i.e. weightings of not necessarily complete graphs will be given with the understanding that edges not mentioned have empty weight.

The following result of Rucinski and Vince [8] is needed for constructions. A graph G of $e(G)$ edges and $v(G)$ vertices is called *balanced* if $e(G)/v(G) \geq e(H)/v(H)$ holds for every subgraph H of G. G is called *strongly balanced* if $e(G)/(v(G)-1) \geq e(H)/(v(H)-1)$ holds for every subgraph H of G. A strongly balanced graph is clearly balanced.

Theorem 2.2 ([8]) *There exists a strongly balanced graph with v vertices and e edges if and only if $1 \leq v-1 \leq e \leq \binom{v}{2}$.* □

Lemma 2.3 *C_n^k is p representable if $p \leq k-2$.*

Proof of Lemma 2.3 We may assume without loss of generality that $p > 2$ by Theorem 1.7. Let $k-1 = a\binom{p+1}{2} + b$ where $0 \leq a$ and $0 \leq b < \binom{p+1}{2}$ are integers. Suppose first, that $b \geq p$. Let G be a balanced graph of $p+1$ vertices and b edges provided by Theorem 2.2. For every $k-1$-element subset of Ω we take K_{p+1} so that edges corresponding to edges of G are weighted by $a+1$-element subsets, the remaining ones by a-element subsets, such that the weights of edges are pairwise disjoint sets, and their union is the given $k-1$-element subset of Ω. We claim that the disjoint union of these weighted complete graphs satisfy the conditions of Lemma 2.1.

It is clear that Condition 1. is satisfied, because weights of adjacent edges are pairwise disjoint sets. Also clear is that every $k-1$-element subset of Ω occurs as union of weights of edges spanned by some $p+1$-element subset of vertices. The only thing to check is that larger subsets of Ω do not occur this way. Let us suppose that the $p+1$-element subset of vertices U is the union of sets U_i, $i = 1, 2, \ldots, t$, where U_i's are the intersections of U with the weighted complete graphs. Let $u_i = |U_i|$, furthermore let e_i be the number of edges of the subgraph of balanced graph G spanned by vertices corresponding to U_i. Then $e_i/u_i \leq b/(p+1)$ is satisfied. The cardinality e of the union of the weights of edges spanned by U can bounded from above, as follows:

$$\begin{aligned} e &\leq a\sum_{i=1}^{t}\binom{u_i}{2} + \sum_{i=1}^{t} e_i \\ &\leq a\binom{p+1}{2} + \sum_{i=1}^{t}\frac{e_i}{u_i}u_i \end{aligned}$$

$$\leq a\binom{p+1}{2} + \sum_{i=1}^{t} \frac{b}{p+1} u_i$$
$$= a\binom{p+1}{2} + b = k-1$$

On the other hand, if $b < p$, then $a > 0$ is satisfied. Let $k-1-p = (a-1)\binom{p+1}{2} + c$. Then $c \geq p$ holds. Let us consider two graphs, G and H, on the same $p+1$ vertices, where G is a balanced graph with c edges, and H is a path (which is clearly balanced). For every $k-1$-element subset of Ω we take K_{p+1} so that edges corresponding to edges of $G \cap H$ are weighted by $a+1$-element subsets, those corresponding to edges of $G \setminus H$ and $H \setminus G$ are weighted by a-element subsets, the remaining ones by $a-1$-element subsets, such that the weights of edges are pairwise disjoint sets, and their union is the given $k-1$-element subset of Ω. That the disjoint union of these weighted complete graphs satisfies the conditions of Lemma 2.1 can be proved by a similar argument to the one above. □

Let us recall that $\lceil x \rceil$ denotes the smallest integer not less than x.

Lemma 2.4 *If*

$$p+1-\left\lceil \frac{p+1}{s} \right\rceil = k-1 \qquad \text{for } s > 1$$

then $p \in \mathrm{SP}(\mathcal{C}_n^k)$

Proof of Lemma 2.4 Take $\binom{n}{s-1}$ paths of s vertices whose edges have one element weights so that each $s-1$-element subset occurs as union of elements of a path. Any $p+1$ vertices span a forest that has at least $\lceil \frac{p+1}{s} \rceil$ components, so at most $k-1$ edges. □

Note, that in Lemma 2.4 $s \leq p+1$ may be assumed. Any $s \geq p+1$ gives the same $p = k-1$ case.

In the following, non-representability of closures is discussed. The general pattern is that a minimal (non-decreasable) representing matrix is assumed, then it is shown that it must contain identical rows that clearly contradicts to its minimality. The next lemma shows that the spectrum of $\mathcal{C}_n^k$ is finite provided n is large enough.

Lemma 2.5 *Let* $p \geq 2k-1$. *If* $n \geq k^2(k-1)$, *then* $\mathcal{C}_n^k$ *is not* p*-representable.*

Proof of Lemma 2.5 Let us assume indirectly that $\mathcal{C}_n^k$ is p-represented by the $m \times n$ matrix M, and M is minimal. Immediately follows that every column has to contain at least $p+1$ pairwise distinct entries, otherwise everything would be (p,p)-dependent on that particular column. According to Lemma 2.1 for every $k-1$-element subset A of Ω there exist $p+1$ vertices of K_m such that the union of weights of edges spanned by these vertices is A. Indeed, A is closed in $\mathcal{C}_n^k$, but cannot be an intersection of other closed sets, because the only closed superset of A is Ω. In particular, for every column $a \in \Omega$ there exists and edge e_a of K_m such that $a \in w(e_a)$. Let $e_1, e_2, \ldots, e_k$ correspond to k distinct columns $\{a_1, a_2, \ldots, a_k\}$.

Suppose, that there exists a column b containing pairwise distinct entries in rows covered by edges e_i . The k edges e_i cover at most $2k \leq p+1$ points, or rows, so there exist $p+1$ points $r_1, r_2, \ldots, r_{p+1}$ such that b contains all different entries in these rows, or in other words: $\bigcup_{i=1}^{k} w(e_i) \subseteq \bigcup_{1 \leq i < j \leq p+1} w(\{r_i, r_j\}) \not\ni b$. This would imply the existence of a closed set of at least k elements which is not Ω, because b is not in the closure of the set $\{a_1, a_2, \ldots, a_k\}$, a contradiction. Thus, each column b must contain at least a pair of identical entries on the at most $2\,k$ rows covered by $e_1, e_2, \ldots, e_k$. Now, $n \geq k^2\,(k-1)$ implies that there are k distinct columns $b_1, b_2, \ldots, b_k$ so that they contain identical elements on the same pair of rows, say r_1, r_2. If there exists a column c containing distinct entries on r_1, r_2, then there exist $p+1$ rows including r_1, r_2 such that c contains all different entries in them, thus a closed set $c \notin B \supseteq \{b_1, b_2, \ldots, b_k\}$ would exist, a contradiction. Consequently, every column must agree on the pair of rows r_1, r_2, i.e., these rows are identical, which contradicts the minimality of M. □

Note that in the above argument the proof of the following proposition is included.

Proposition 2.6 *If the matrix M p-represents C_n^k and minimal subject this condition, then the weight of an edge $w(e)$ is at most $k-1$-element set.*

The next proposition considers another property of a minimal representation.

Proposition 2.7 *Let $p \leq 2\,k-4$ and $n \geq (k-1)\,(2k-3)$. Let M p-represent C_n^k and let M be minimal subject to this condition. Then for any $p+1$ rows $r_1, r_2, \ldots, r_{p+1}$,*

$$\left| \bigcup_{1 \leq i < j \leq p+1} w(\{r_i, r_j\}) \right| \leq k-1.$$

Proof of Proposition 2.7 According to Lemma 2.1 the union of edge weights of a $p+1$-point complete subgraph is either Ω or its size is at most $k-1$. Suppose indirectly, that there is a sub-K_{p+1} P such that the union of its edge weights is Ω. M p-represents C_n^k, so there is a sub-K_{p+1} Q such that the union of its edge weights is a $k-1$-element subset. By successively shifting vertices from $P \setminus Q$ to Q, sub-K_{p+1} P' and Q' are obtained that $|P' \setminus Q'| = 1$, but the union of edge weights of P' is still Ω, while that of Q' is still a $k-1$-element subset. Let $\{b\} = P' \setminus Q'$. Then the union of edge weights of the p edges between b and $P' \cap Q'$ is of size at least $n-k+1$, thus there exists an edge e amongst them such that $|w(e)| \geq k$, that contradicts to the minimality of M by Proposition 2.6. □

The next proposition allows considering p-representations of special type.

Proposition 2.8 *Let $2\,\lfloor \frac{p+1}{2} \rfloor \geq k$ and suppose that C_n^k is p-representable. Suppose furthermore that $p \leq 2\,k-4$ and $n \geq (k-1)\,(2k-3)$. Then there exists $n' \geq n-k+1$ such that $C_{n'}^k$ is p-represented so that each edge weight is at most one element set.*

Proof of Proposition 2.8 Let M be a matrix p-representing C_n^k that is minimal subject to this condition. A sequence of $\lfloor \frac{p+1}{2} \rfloor$ edges is defined . Let a_1 be the largest size of an edge weight, and let e_1 be an edge of weight of this size. Now suppose, that $e_1, e_2, \ldots, e_i$ are already defined and let a_{i+1} be the maximum of $|w(e) \setminus \cup_{j \le i} w(e_j)|$ for any edge of K_m and define e_{i+1} to be an edge attaining this maximum. We claim, that $a_{\lfloor \frac{p+1}{2} \rfloor} = 1$. Indeed, otherwise $|\cup_{i=1}^{\lfloor \frac{p+1}{2} \rfloor} w(e_i)| \ge k$ would be, which contradicts to Proposition 2.7, because any $\lfloor \frac{p+1}{2} \rfloor$ edges can be embedded into a sub-K_{p+1}. Let $\Omega_1 = \Omega \setminus \cup_{i=1}^{\lfloor \frac{p+1}{2} \rfloor} w(e_i)$. Then $|\Omega_1| \ge n - k + 1$ and M restricted to the columns of Ω_1 p-represents $C_{|\Omega_1|}^k$ with the property, that each edge of K_m has weight of size at most one. □

The next lemma is a sort of converse of Lemma 2.4.

Lemma 2.9 *Let $n \ge (k-1)(2k-3)$ and suppose that there exists integer $s > 1$ such that*

$$p + 1 - \left\lceil \frac{p+1}{s} \right\rceil < k - 1 < p + 1 - \left\lceil \frac{p+1}{s+1} \right\rceil .$$

Then C_n^k is not *p-representable.*

Proof of Lemma 2.9 Let us suppose indirectly that C_n^k is p-represented by $m \times n$ matrix M. We may assume without loss of generality that each edge weight of K_m is at most one element set according to Proposition 2.8. In the following "number of edges" means "number of edges of pairwise different weights" for the sake of simplicity. If there are more than one edges of the same non-empty weight in a sub-K_{p+1}, then an arbitrary one of them can be picked.

Each $k-1$-element subset of Ω must occur as union of weights of edges of a sub-K_{p+1}. By the condition on k and p, the edges of non-empty but pairwise different weight of such a sub-K_{p+1} span a graph that has a non-tree component or a tree component of size at least $s+1$. Such a component is called *big*. Let $B_1, B_2, \ldots, B_z$ be big components of different sub-K_{p+1}'s corresponding to pairwise disjoint $k-1$-element subsets. A $p+1$- vertices subgraph is constructed as follows. First, take as many non-tree components as possible, then big tree components, until the number of vertices reaches $p+1$. Let this graph be H, and suppose the number of vertices of H covered by non-tree components is d, and let $u = p+1-d$. Then the number of edges $e(H)$ of H satisfies

$$e(H) \ge d + u + \left\lceil \frac{u}{s+1} \right\rceil \ge p + 1 - \left\lceil \frac{p+1}{s+1} \right\rceil > k - 1,$$

that contradicts to Proposition 2.7. □

The above results can be summarized in the following theorem.

Theorem 2.10 *Let $n \ge k^2(k-1)$. Then the spectrum $\mathrm{SP}(C_n^k)$ of C_n^k is determined by the following formula:*

$$\mathrm{SP}(C_n^k) = \{1, 2, \ldots, k-1\} \cup \{p : \exists s \in \mathbf{N}\ p + 1 - \left\lceil \frac{p+1}{s} \right\rceil = k - 1\}.$$

□

3 Open Problems

A complete characterization of $\mathrm{SP}(C_n^k)$, was given if k is small with respect to n. However, it was proved in [6] that C_n^n is p- representable for every positive integer p. Thus, the following problem arises naturally.

Open Problem 1 *Determine those k's for which $\mathrm{SP}(C_n^k) = \mathbf{N}$ holds!*

The constructions used in proving that certain values of p are in the spectrum of C_n^k usually result in very large matrices. Thus, the next problem is also of interest. For similar results and problems the reader is referred to [6].

Open Problem 2 *Determine the minimum number of rows of a matrix p-representing C_n^k, provided such a representation exists!*

Finally, the general question is still open.

Open Problem 3 *Determine the spectra of other closures!*

Open Problem 3 is in particular interesting for closures arising in different areas of combinatorics, for example for closures coming from matroids.

References

[1] W.W. ARMSTRONG, Dependency Structures of database Relationships, *Information Processing 74* (North Holland, Amsterdam, 1974) 580-583.

[2] E.F. CODD, A Relational Model of Data for Large Shared Data Banks, *Comm. ACM,* **13** (1970) 377-387.

[3] J. DEMETROVICS, G.O.H. KATONA, Extremal combinatorial problems in a relational database, in: *Fundamentals of Computation Theory 81, Proc. 1981 Int. FCT-Conf.*, Szeged, Hungary, 1981, Lecture Notes in Computer Science 117 (Springer, Berlin 1981) pp. 110-119.

[4] J. DEMETROVICS, G.O.H. KATONA, A survey of some combinatorial results concerning functional dependencies in database relations, *Annals of Math. and Artificial Intelligence* **7** (1993) 63-82.

[5] J. DEMETROVICS, G.O.H. KATONA AND A.SALI, The characterization of branching dependencies, *Discrete Appl. Math.*, **40** (1992), 139-153.

[6] J. DEMETROVICS, G.O.H. KATONA AND A. SALI, Minimal Representations of Branching Dependencies, *Acta Sci. Math. (Szeged)*, **60**, (1995) 213-223.

[7] J. DEMETROVICS, G.O.H. KATONA AND A.SALI, Design Type Problems Motivated by Database Theory, submitted.

[8] A. RUCINSKI AND A. VINCE, Strongly balanced graphs and random graphs, *J. Graph Theory* **10** (1986), 251-264.

Received January, 1998

Acta Cybernetica **13** (1998) 439-452.

Analysis of the Completion Time of Markov Reward Models and its Application*

Miklós Telek † András Pfening † Gábor Fodor †

Abstract

Analysis of Markov Reward Models (*MRM*) with preemptive resume (prs) policy usually results in a double transform expression, whose solution is based on the inverse transformations both in time and reward variable domain. This paper discusses the case when the reward rates can be either 0 or positive, and analyses the completion time of *MRM*s. We present a symbolic expression of moments of the completion time, from which a computationally effective recursive numerical method can be obtained. As a numerical example the mean and the standard deviation of the completion time of a Carnegie-Mellon multiprocessor system are evaluated by the proposed method.

Key words: Markov Reward Models, Preemptive Resume Policy, Completion Time.

1 Introduction

The properties of stochastic reward processes have been studied since a long time [9]. However, only recently, stochastic reward models (*SRM*) have received attention as a modeling tool in performance and reliability evaluation. Indeed, the possibility of associating a reward variable to each structure state increases the descriptive power and the flexibility of the model.

Different interpretations of the structure-state process and of the associated reward structure give rise to different applications. Common assignments of the reward rates are: execution rates of tasks in computing systems (the computational capacity) [1], number of active processors (or processing power), throughput [12], available bandwidth average response time or response time distribution.

Two main different points of view have been assumed in the literature when dealing with *SRM* for degradable systems [11]. In the *system oriented* point of view the most significant measure is the total amount of work done by the system in a finite interval. The accumulated reward is a random variable whose distribution function is sometimes called *performability* [12]. Various numerical techniques for

*A. Pfening and M. Telek thank Hungarian OTKA for grant No. F-23971.

†Department of Telecommunications and Telematics, Technical University of Budapest, P.O.B. 91, 1521 Budapest, Hungary

the evaluation of the performability have been investigated in recent papers: [10, 7, 8]. In the *user oriented* (or *task oriented*) point of view the system is regarded as a server, and the emphasis of the analysis is on the ability of the system to accomplish an assigned task in due time. Consequently, the most characterizing measure becomes the probability of accomplishing an assigned service in a given time. The task oriented point of view is a more direct representation of the quality of service. Asymptotic behaviour of some task oriented measures is studied in [16] under the assumption of fast service (or repair).

A unified formulation to the system oriented and the user oriented point of view was provided by Kulkarni et al. in [11]. An alternative interpretation of the completion time problem can be given in terms of the hitting time of an appropriate cumulative functional [6] against an absorbing barrier equal to the work requirement. The definition of a cumulative functional was first suggested by Kulkarni et al. [11] and then explicitly exploited in [4], where the completion time was modelled as a first hitting time against an absorbing barrier. The subclass of *MRM*s with Phase-type distributed random work requirement was studied by Bobbio and Trivedi [5]. In this case the completion time is Phase type distributed and they defined the "extended" Continuous Time Markov Chain (*CTMC*) which characterize the distribution of the completion time.

In this paper, we improve the results of [2, 13] and propose a computationally effective approach not only to calculate the mean completion time of on-off *MRM*s (i.e. *MRM*s with reward rates equal to 0 or 1), but to obtain all the moments of the completion time of *MRM*s with arbitrary non-negative reward rates.

The paper is organized as follows. Section 2 provides the formal definition of *SRM*s, and introduces the class of *MRM*s. In Section 3 the completion time analysis of *MRM*s is presented. Section 4 gives an application of the proposed computational approach to the task completion time analysis of a Carnegie-Mellon multiprocessor system. The paper is concluded in Section 5.

2 Stochastic Reward Models

The adopted modeling framework consists of describing the behaviour of the system configuration in time by means of a stochastic process, and by associating a non-negative real constant to each state of the structure-state process representing the effective working capacity or performance level or cost or stress of the system in that state. The variable associated to each structure-state is called the *reward rate* [9].

Let the *structure-state process* $Z(t)$ $(t \geq 0)$ be a (right continuous) stochastic process defined over a discrete and finite state space Ω of cardinality n. Let f be a non-negative real-valued function defined as:

$$f[Z(t)] = r_i \geq 0 \ , \qquad \text{if } Z(t) = i \ , \tag{1}$$

$f[Z(t)]$ represents the instantaneous reward rate associated to state i.

Definition 1 *The* **accumulated reward** $B(t)$ *is a random variable which represents the accumulation of reward in time:*

$$B(t) = \int_0^t f[Z(\tau)]d\tau = \int_0^t r_{Z(\tau)}d\tau.$$

$B(t)$ is a stochastic process that depends on $Z(u)$ for $0 \leq u \leq t$. According to Definition 1 this paper restricts the attention to the class of models in which no state transition can entail to a loss of the accumulated reward. A *SRM* of this kind is called *preemptive resume* (prs) model. The distribution of the accumulated reward is defined as

$$B(t,w) = Pr\{B(t) \leq w\}.$$

The complementary question concerning the reward accumulation of *SRMs* is the time needed to complete a given (possibly random) work requirement (i.e., the time to accumulate the required amount of reward).

Definition 2. *The* **completion time** C *is the random variable representing the time to accumulate a reward requirement equal to a random variable* W*:*

$$C = min\,[t \geq 0 : B(t) = W] .$$

C is the time instant at which the work accumulated by the system reaches the value W for the first time. Assume, in general, that W is a random variable independent from $Z(t)$ with distribution $G(w)$ with support on $(0, \infty)$. The degenerate case, in which W is deterministic and the distribution $G(w)$ becomes the unit step function $U(w - w_d)$, can be considered as well. For a given sample of $W = w$, the completion time $C(w)$ and its cumulative distribution function $C(t,w)$ are defined as:

$$C(w) = \min\,[t \geq 0 : B(t) = w] \;; \quad C(t,w) = Pr\,\{C(w) \leq t\} . \tag{2}$$

The completion time C is characterized by the following distribution:

$$\hat{C}(t) = Pr\,\{C \leq t\} = \int_0^\infty C(t,w)\,dG(w) . \tag{3}$$

The distribution of the completion time of a *prs SRM* is closely related to the distribution of the accumulated reward by means of the following relation:

$$B(t,w) = Pr\,\{B(t) \leq w\} = Pr\,\{C(w) \geq t\} = 1 - C(t,w) . \tag{4}$$

For the purposes of the subsequent analysis below we define the following matrix functions $\mathbf{P}(t,w) = \{P_{ij}(t,w)\}$ and $\mathbf{F}(t,w) = \{F_{ij}(t,w)\}$ as:

$$P_{ij}(t,w) = Pr\{Z(t) = j\,,\, B(t) \leq w \,|\, Z(0) = i\}\,, \tag{5}$$

$$F_{ij}(t,w) = Pr\{Z(C(w)) = j\,,\, C(w) \leq t \,|\, Z(0) = i\}\,, \tag{6}$$

- $P_{ij}(t,w)$ is the joint distribution of the accumulated reward and the structure state at time t supposed that the initial state of the structure state process is i,
- $F_{ij}(t,w)$ is the joint distribution of the completion time and the structure state at completion supposed that the initial state of the structure state process is i.

We assume (5) and (6), it follows for any t and i that $\sum_{j\in\Omega}[P_{ij}(t,w)+F_{ij}(t,w)] = 1$.
By these definitions:

$$B(t,w) = \underline{P}(0)\,\mathbf{P}(t,w)\,\underline{h}^T \quad \text{and} \quad C(t,w) = \underline{P}(0)\,\mathbf{F}(t,w)\,\underline{h}^T ,$$

where $\underline{P}(0) = \{P_i(0)\}$ is the row vector of the initial state probabilities of the structure-state process ($Pr\{Z(0)=i\} = P_i(0)$), and $\underline{h}^T$ is the column vector with all the entries equal to 1.

Given that $G(w)$ is the cumulative distribution function of the random work requirement W, the distribution of the completion time is:

$$\hat{C}(t) = \int_{w=0}^{\infty} \left[\sum_{i\in\Omega}\sum_{j\in\Omega} P_i(0)F_{ij}(t,w) \right] dG(w) = \int_{w=0}^{\infty} \underline{P}(0)\,\mathbf{F}(t,w)\,\underline{h}^T\, dG(w) . \tag{7}$$

2.1 Markov Reward Models

Definition 3. *The subclass of SRMs in which the structure state process ($Z(t)$) is an ergodic* CTMC *with any initial probability distribution is called Markov Reward Models (*MRM*).*

The introduced matrix functions of a *MRM* can be described in double transform domain based on the infinitesimal generator ($\mathbf{A}$) of the structure state process and the reward rates. Detailed derivations presented in [11, 17] results in:

$$F^{\sim *}_{ij}(s,v) = \delta_{ij}\,\frac{r_i}{s+vr_i-a_{ii}} + \sum_{k\in R,k\neq i}\frac{a_{ik}}{s+vr_i-a_{ii}}F^{\sim *}_{kj}(s,v) \tag{8}$$

$$P^{\sim *}_{ij}(s,v) = \delta_{ij}\,\frac{s}{v(s+vr_i-a_{ii})} + \sum_{k\in R,k\neq i}\frac{a_{ik}}{s+vr_i-a_{ii}}P^{\sim *}_{kj}(s,v) \tag{9}$$

where δ_{ij} is the Kronecker delta.

The final expressions take the following matrix forms:

$$\mathbf{F}^{\sim *}(s,v) = (s\mathbf{I}+v\mathbf{R}-\mathbf{A})^{-1}\mathbf{R} \tag{10}$$

$$\mathbf{P}^{\sim *}(s,v) = \frac{s}{v}\,(s\mathbf{I}+v\mathbf{R}-\mathbf{A})^{-1} \tag{11}$$

where $\sim$ denotes the Laplace-Stieltjes transform with respect to $t(\rightarrow s)$, * denotes the Laplace transform with respect to $w(\rightarrow v)$, $\mathbf{I}$ is the identity matrix and $\mathbf{R}$ is the diagonal matrix of the reward rates (r_i); the dimensions of $\mathbf{I}$, $\mathbf{R}$, $\mathbf{A}$, $\mathbf{F}$ and $\mathbf{P}$ are $(n \times n)$.

Starting from equations (10-11), the evaluation of the reward measures of a *MRM* requires the following steps:

1. symbolic evaluation of the entries of the $\mathbf{P}^{*\sim}(s,v)$ and $\mathbf{F}^{*\sim}(s,v)$ matrices in the double transform domain according to (10) and (11), which requires a symbolic inversion of an $n \times n$ size matrix;

2. symbolic inverse Laplace-Stieltjes transformation of $\mathbf{P}^{*\sim}(s,v)$ and/or $\mathbf{F}^{*\sim}(s,v)$ with respect to s;

3. numerical inverse Laplace transformation with respect to v;

4. unconditioning of the result by a numerical integration according to the distribution of the work requirement defined by (7).

However, this way of the analysis contains some computationally intensive steps, and the whole procedure can be applied to very small scale problems (less than 6-8 states) only.

3 Completion time analysis of *MRM*s

According to the associated reward rates the states of *MRM*s can be divided into two parts, namely S and $S^c = \Omega - S$, where S contains the states with positive reward rates and S^c with zero reward rates, i.e., $\forall i \in S$, $r_i > 0$ and $\forall i \in S^c$, $r_i = 0$. Suppose that S contains m states out of n. Thus we can renumber the states in Ω in a way that the states numbered $1, 2, \ldots, m$ belong to S and the states numbered $m+1, m+2, \ldots, n$ belong to S^c. By this ordering of the states, $\mathbf{A}$ can be partitioned into the following form $\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix}$, where $\mathbf{A}_1$ describes the transitions inside S, $\mathbf{A}_2$ contains the intensity of the transitions from S to S^c, $\mathbf{A}_3$ the transitions from S^c to S, and $\mathbf{A}_4$ the transitions inside S^c. Note that according to Definition 3 $Z(t)$ is an ergodic *CTMC*, hence the completion time of a finite work requirement w is finite with probability 1 and $\mathbf{A}_4^{-1}$ exists. By the renumbering of states the diagonal matrix of the reward rates has the form $\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, where $\mathbf{R}_1 = \mathrm{Diag}_{i \in S} < r_i >$ is the diagonal matrix of the reward rates in S with cardinality $m \times m$.

3.1 Moments of the completion time of *MRM*s

In this section we calculate the moments of the completion time using the Laplace-Stieltjes transform, and we propose a recursive method to calculate the moments

in a computationally effective way. We make use of the idea proposed by Iyer et al. for the analysis of the accumulated reward [10]. The nth moment of the completion time of w amount of work is defined by

$$M_{(n)}(w) = E\{C(w)^n\} = \int_{t=0}^{\infty} t^n \, d\, C(t,w) \; .$$

Theorem 1. *The nth moment of the completion time of an* MRM *with work requirement w is:*

$$M_{(n)}(w) = n! \; \underline{P}(0) \; \mathrm{LT}^{-1}\left[(\mathbf{R}v - \mathbf{A})^{-(n+1)}\mathbf{R}\right] \; \underline{h}^T \tag{12}$$

where LT^{-1} *means the inverse Laplace transformation with respect to* v.

Proof: The moments can be calculated using the Laplace-Stieltjes transform of the completion time and substituting equation (10):

$$\begin{aligned} M_{(n)}(w) &= (-1)^n \left.\frac{\partial^n \mathrm{LT}^{-1}\left[C^{\sim *}(s,v)\right]}{\partial s^n}\right|_{s=0} \\ &= (-1)^n \left.\frac{\partial^n \mathrm{LT}^{-1}\left[\underline{P}(0)\; \mathbf{F}^{\sim *}(s,v)\; \underline{h}^T\right]}{\partial s^n}\right|_{s=0} \\ &= (-1)^n \underline{P}(0) \left.\frac{\partial^n \mathrm{LT}^{-1}\left[\mathbf{F}^{\sim *}(s,v)\right]}{\partial s^n}\, \underline{h}^T\right|_{s=0} \\ &= (-1)^n \underline{P}(0) \left.\frac{\partial^n \mathrm{LT}^{-1}\left[(s\mathbf{I} + v\mathbf{R} - \mathbf{A})^{-1}\mathbf{R}\right]}{\partial s^n}\right|_{s=0} \underline{h}^T \; . \end{aligned} \tag{13}$$

We assume in the above formula that the order of the inversion and the derivation can be changed:

$$M_{(n)}(w) = (-1)^n \underline{P}(0)\, \mathrm{LT}^{-1}\left[\left.\frac{\partial^n (s\mathbf{I} + v\mathbf{R} - \mathbf{A})^{-1}\mathbf{R}}{\partial s^n}\right|_{s=0}\right] \underline{h}^T .$$

The derivation can be accomplished using Leibniz's rule, and setting the value of s to 0:

$$M_{(n)}(w) = n! \; \underline{P}(0) \; \mathrm{LT}^{-1}\left[(v\mathbf{R} - \mathbf{A})^{-(n+1)}\mathbf{R}\right] \; \underline{h}^T .$$

□

3.2 Analysis of the mean completion time of *MRM*s

Because of the inverse Laplace transformation and matrix inversion contained in equation (12) the calculation of the moments is a computationally intensive task.

Begain et al. [2] proposed an effective method to calculate the first moment, i.e., the mean value of the completion time of on-off reward models. Here we generalize that result for the mean completion time of *MRM*s with general reward structure.

Theorem 2. *The expected time while a* MRM *with general reward rates completes* w *amount of work is:*

$$E\{C(w)\} = \underline{P}(0) \begin{bmatrix} \mathbf{L}(w) & -\mathbf{L}(w)\mathbf{A}_2\mathbf{A}_4^{-1} \\ -\mathbf{A}_4^{-1}\mathbf{A}_3\mathbf{L}(w) & -\mathbf{A}_4^{-1} + \mathbf{A}_4^{-1}\mathbf{A}_3\mathbf{L}(w)\mathbf{A}_2\mathbf{A}_4^{-1} \end{bmatrix} \underline{h}^T \quad , \quad (14)$$

where

$$\mathbf{L}(w) = \int_0^w e^{u\mathbf{R}_1^{-1}\boldsymbol{\beta}} du\ \mathbf{R}_1^{-1} \quad \textit{and} \quad \boldsymbol{\beta} = \mathbf{A}_1 - \mathbf{A}_2\mathbf{A}_4^{-1}\mathbf{A}_3.$$

Proof:

$$\begin{aligned} E\{C(w)\} &= \int_{t=0}^{\infty} (1 - C(t,w))\, dt = \int_{t=0}^{\infty} B(t,w)\, dt \\ &= \lim_{s\to 0} \frac{1}{s} B^{\sim}(s,w) = \lim_{s\to 0} \underline{P}(0)^T \mathbf{P}^{\sim}(s,w)\underline{h}^T \qquad (15) \\ &= \underline{P}(0)\ \mathrm{LT}^{-1}\left[\frac{1}{v}\,(v\mathbf{R} - \mathbf{A})^{-1}\right] \underline{h}^T \ . \end{aligned}$$

Let us consider the term $\mathrm{LT}^{-1}\left[\frac{1}{v}\,(v\mathbf{R} - \mathbf{A})^{-1}\right]$ separately using the partitioned form $\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$.

$$\mathrm{LT}^{-1}\left[\frac{1}{v}\,(v\mathbf{R} - \mathbf{A})^{-1}\right] = \mathrm{LT}^{-1}\left\{\frac{1}{v}\begin{bmatrix} v\mathbf{R}_1 - \mathbf{A}_1 & -\mathbf{A}_2 \\ -\mathbf{A}_3 & -\mathbf{A}_4 \end{bmatrix}^{-1}\right\}$$

$$= \mathrm{LT}^{-1}\left\{\frac{1}{v}\begin{bmatrix} (v\mathbf{R}_1 - \boldsymbol{\beta})^{-1} & -(v\mathbf{R}_1 - \boldsymbol{\beta})^{-1}\mathbf{A}_2\mathbf{A}_4^{-1} \\ -\mathbf{A}_4^{-1}\mathbf{A}_3(v\mathbf{R}_1 - \boldsymbol{\beta})^{-1} & \mathbf{A}_4^{-1} + \mathrm{A}_4{}^{-1}\mathrm{A}_3(v\mathbf{R}_1 - \boldsymbol{\beta})^{-1}\mathbf{A}_2\mathbf{A}_4^{-1} \end{bmatrix}\right\}$$

$$= \mathrm{LT}^{-1}\left\{\frac{1}{v}\begin{bmatrix} (v\mathbf{I}_1 - \mathbf{R}_1^{-1}\boldsymbol{\beta})^{-1}\mathbf{R}_1^{-1} & -(v\mathbf{I}_1 - \mathbf{R}_1^{-1}\boldsymbol{\beta})^{-1}\mathbf{R}_1^{-1}\mathbf{A}_2\mathbf{A}_4^{-1} \\ -\mathbf{A}_4^{-1}\mathbf{A}_3(v\mathbf{I}_1 - \mathbf{R}_1^{-1}\boldsymbol{\beta})^{-1}\mathbf{R}_1^{-1} & \mathbf{A}_4^{-1} + \mathrm{A}_4{}^{-1}\mathrm{A}_3(v\mathbf{I}_1 - \mathbf{R}_1^{-1}\boldsymbol{\beta})^{-1}\mathbf{R}_1^{-1}\mathbf{A}_2\mathbf{A}_4^{-1} \end{bmatrix}\right\}$$

$$= \begin{bmatrix} \mathbf{L}(w) & -\mathbf{L}(w)\mathbf{A}_2\mathbf{A}_4^{-1} \\ -\mathbf{A}_4^{-1}\mathbf{A}_3\mathbf{L}(w) & \mathbf{A}_4^{-1} + \mathbf{A}_4^{-1}\mathbf{A}_3\mathbf{L}(w)\mathbf{A}_2\mathbf{A}_4^{-1} \end{bmatrix} . \qquad (16)$$

In (16), the first step is the partitioned description based on the block structure of matrices $\mathbf{A}$ and $\mathbf{R}$; the second step is the application of the inverse of a partitioned matrix ([3, 14]); the third step comes as the product of inverse matrices ([3, 14]); while the fourth step is because the Laplace transform of $\mathbf{L}(w)$ has the form

$$\mathbf{L}^*(v) = \frac{1}{v} \left(v\mathbf{I}_1 - \mathbf{R}_1^{-1}\beta\right)^{-1} \mathbf{R}_1^{-1}.$$

From (16) the theorem follows. □

An intuitive proof of Theorem 2 is possible based on the interpretation of matrix β. Define $Z'(t')$ a *CTMC* over S based on the original structure state process $Z(t) \in \Omega$ as follows:

$$Z'(t') = Z(t); \qquad \frac{dt'}{dt} = 1; \qquad \text{if } Z(t) \in S\ ,$$

$$\frac{dt'}{dt} = 0; \qquad \text{if } Z(t) \in \Omega - S\ ,$$

i.e., $Z'(t')$ takes the same state as $Z(t)$ when $Z(t) \in S$ and the clock t' is switched on (off) when $Z(t) \in S$ ($Z(t) \in \Omega - S$). β is the infinitesimal generator of *CTMC* $Z'(t')$ over S (with the usual properties: $\forall i, j \in S$, $\beta_{ij}|_{i \neq j} \geq 0$ and $\sum_{j \in S} \beta_{ij} = 0$). The multiplication with $\mathbf{R}_1^{-1}$ stands for scaling and rescaling the time providing a constant reward increment rate as proposed by Beaudry [1]. $Z'(t')$ is the stochastic process which characterize the reward accumulation as captured by $\mathbf{L}(w)$. The submatrices in (14) account for the time $Z(t)$ spends out of S.

3.3 A recursive analysis of higher moments

Here we propose a recursive method to calculate the higher moments. First we introduce some notation. Let $M_{ij\,(n)}(w)$ be the nth moment of the completion time assuming that the process was started in state i, the work requirement was completed in state j and the work requirement was w. Let $\mathbf{M}_{(n)}(w)$ be a matrix with entries $M_{ij\,(n)}(w)$, and $\mathbf{M}^*_{(n)}(v)$ be the Laplace transform of $\mathbf{M}_{(n)}(w)$. Let

$$\mathbf{F}^{\sim *\,(n)}(0, v) = \left.\frac{\partial^n \mathbf{F}^{\sim *}(s, v)}{\partial s^n}\right|_{s=0}.$$

Theorem 3. *The nth moment ($n \geq 2$) of the completion time of an* MRM *with work requirement w can be obtained as*

$$\begin{aligned} M_{(n)}(w) &= \underline{P}(0)\, \mathbf{M}_{(n)}(w)\, \underline{h}^T \\ &= n\underline{P}(0) \int_{y=0}^{w} \Theta(w-y)\, \mathbf{M}_{(n-1)}(y)\, \underline{h}^T\, dy + n\, \underline{P}(0)\, \hat{\mathbf{A}}\, \mathbf{M}_{(n-1)}(w)\, \underline{h}^T, \end{aligned} \tag{17}$$

where

$$\Theta(w) = \begin{bmatrix} e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}\mathbf{R}_1^{-1}} & -e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}\mathbf{R}_1^{-1}}\mathbf{A}_2\mathbf{A}_4^{-1} \\ -\mathbf{A}_4^{-1}\mathbf{A}_3 e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}\mathbf{R}_1^{-1}} & \mathbf{A}_4^{-1}\mathbf{A}_3 e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}\mathbf{R}_1^{-1}}\mathbf{A}_2\mathbf{A}_4^{-1} \end{bmatrix}$$

$$\textit{and} \quad \hat{\mathbf{A}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{A}_4^{-1} \end{bmatrix}.$$

Proof: From equation (10)

$$(s\mathbf{I} + v\mathbf{R} - \mathbf{A})\mathbf{F}^{\sim *}(s,v) = \mathbf{R} \ . \tag{18}$$

Using Leibniz's rule, the differentiation of equation (18) $n+1$ times with respect to s and setting $s = 0$ yields

$$\mathbf{F}^{\sim *\,(n+1)}(0,v) = -(n+1)(\mathbf{R}v - \mathbf{A})^{-1}\mathbf{F}^{\sim *\,(n)}(0,v) \ . \tag{19}$$

Because $\mathbf{M}^*_{(n)}(v) = (-1)^n\mathbf{F}^{\sim *\,(n)}(0,v)$ according to equation (13), equation (19) can be rewritten as

$$\mathbf{M}^*_{(n+1)}(v) = (n+1)\,(\mathbf{R}v - \mathbf{A})^{-1}\mathbf{M}^*_{(n)}(v). \tag{20}$$

Let us consider the term $\mathrm{LT}^{-1}\left[(v\mathbf{R} - \mathbf{A})^{-1}\right]$ separately.

$$\mathrm{LT}^{-1}\left[(v\mathbf{R} - \mathbf{A})^{-1}\right] = \mathrm{LT}^{-1}\left\{\begin{bmatrix} v\mathbf{R}_1 - \mathbf{A}_1 & -\mathbf{A}_2 \\ -\mathbf{A}_3 & -\mathbf{A}_4 \end{bmatrix}^{-1}\right\}$$

$$= \mathrm{LT}^{-1}\begin{bmatrix} (v\mathbf{R}_1 - \boldsymbol{\beta})^{-1} & (v\mathbf{R}_1 - \boldsymbol{\beta})^{-1}\mathbf{A}_2\mathbf{A}_4^{-1} \\ -\mathbf{A}_4^{-1}\mathbf{A}_3(v\mathbf{R}_1 - \boldsymbol{\beta})^{-1} & \mathbf{A}_4^{-1} + \mathbf{A}_4^{-1}\mathbf{A}_3(v\mathbf{R}_1 - \boldsymbol{\beta})^{-1}\mathbf{A}_2\mathbf{A}_4^{-1} \end{bmatrix}$$

$$= \mathrm{LT}^{-1}\begin{bmatrix} (v\mathbf{I}_1 - \mathbf{R}_1^{-1}\boldsymbol{\beta})^{-1}\mathbf{R}_1^{-1} & (v\mathbf{I}_1 - \mathbf{R}_1^{-1}\boldsymbol{\beta})^{-1}\mathbf{R}_1^{-1}\mathbf{A}_2\mathbf{A}_4^{-1} \\ -\mathbf{A}_4^{-1}\mathbf{A}_3(v\mathbf{I}_1 - \mathbf{R}_1^{-1}\boldsymbol{\beta})^{-1}\mathbf{R}_1^{-1} & \mathbf{A}_4^{-1} + \mathbf{A}_4^{-1}\mathbf{A}_3(v\mathbf{I}_1 - \mathbf{R}_1^{-1}\boldsymbol{\beta})^{-1}\mathbf{R}_1^{-1}\mathbf{A}_2\mathbf{A}_4^{-1} \end{bmatrix}$$

$$= \begin{bmatrix} e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}\mathbf{R}_1^{-1}} & e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}\mathbf{R}_1^{-1}}\mathbf{A}_2\mathbf{A}_4^{-1} \\ -\mathbf{A}_4^{-1}\mathbf{A}_3 e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}\mathbf{R}_1^{-1}} & \mathbf{A}_4^{-1}\,\delta(w) + \mathbf{A}_4^{-1}\mathbf{A}_3 e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}\mathbf{R}_1^{-1}}\mathbf{A}_2\mathbf{A}_4^{-1} \end{bmatrix}. \tag{21}$$

The steps in (21) are similar to the steps in (16); the only difference is that here we have the inverse Laplace transform of $(v\mathbf{I}_1 - \mathbf{R}_1^{-1}\boldsymbol{\beta})^{-1}\,\mathbf{R}_1^{-1}$ which is $e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}\mathbf{R}_1^{-1}}$.

Hence $\mathrm{LT}^{-1}\left[\,(v\mathbf{R}-\mathbf{A})^{-1}\right] = \Theta(w) + \hat{\mathbf{A}}\delta(w)$, where $\delta(w)$ denotes the Dirac delta function (a Dirac impulse at $w=0$), the inversion and the integration yields the theorem. □

To apply the result of Theorem 3 for the evaluation of the first moment we shall define in accordance with equation (13)

$$M_{(0)}(w) = \mathrm{LT}^{-1}\left[C^{\sim *}(0,v)\right] = \mathrm{LT}^{-1}\left[\underline{P}(0)\ \mathbf{F}^{\sim *}(0,v)\ \underline{h}^T\right]$$

and

$$\mathbf{M}^*_{(0)}(v) = \mathbf{F}^{\sim *}(0,v).$$

To express the first moment we use equation (17) and then equation (20) to obtain

$$\begin{aligned} M_{(1)}(w) &= \mathrm{LT}^{-1}\left[\underline{P}(0)\ \mathbf{M}^*_{(1)}(v)\ \underline{h}^T\right] \\ &= \mathrm{LT}^{-1}\left[\underline{P}(0)\ (\mathbf{R}v-\mathbf{A})^{-1}\,\mathbf{M}^*_{(0)}(v)\ \underline{h}^T\right], \end{aligned}$$

which is by definition

$$\begin{aligned} M^{(1)}(w) &= \mathrm{LT}^{-1}\left[\underline{P}(0)\ (\mathbf{R}v-\mathbf{A})^{-1}\,\mathbf{F}^{\sim *}(0,v)\ \underline{h}^T\right] \\ &= \mathrm{LT}^{-1}\left[\underline{P}(0)\ (\mathbf{R}v-\mathbf{A})^{-2}\,\mathbf{R}\ \underline{h}^T\right] \\ &= \mathrm{LT}^{-1}\left[\underline{P}(0)\ \frac{1}{v}\,(\mathbf{R}v-\mathbf{A})^{-1}\ \underline{h}^T\right], \end{aligned}$$

since $(\mathbf{R}v-\mathbf{A})^{-2}\,\mathbf{R}\ \underline{h}^T = 1/v\ (\mathbf{R}v-\mathbf{A})^{-1}\ \underline{h}^T$, because $\mathbf{A}\,\underline{h}^T = \underline{0}^T$. The inverse transform gives the result of Theorem 2.

If the system is started from operational states, which is a rather realistic assumption, (i.e., $\forall i \in S^c, P_i(0)=0$), then one can neglect the second term of equation (17). This term stands for the time needed to start the reward accumulation (i.e., to enter S) when the system starts from S^c.

Another important analysis problem of *MRMs* is the probability distribution of the structure state process at completion, i.e., $P^c_{ij} = Pr\{Z(C)=j | Z(0)=i\}$. For example, the required maintenance after a mission of a system which started from a particular state can be estimated based on this performance measure. A closed form expression of the probability distribution at completion, by which its effective computation is possible, comes by the following theorem.

Theorem 4. *The probability of being in state j at completion given that the process started from state i can be computed as follows:*

$$P^c_{ij} = \int_{w=0}^{\infty} \begin{bmatrix} e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}} & \mathbf{0} \\ -\mathbf{A}_4^{-1}\mathbf{A}_3 e^{w\mathbf{R}_1^{-1}\boldsymbol{\beta}} & \mathbf{0} \end{bmatrix}_{ij} dG(w) \qquad (22)$$

Proof: By the known transform domain measures we have:

$$\begin{aligned} P_{ij}^{c} &= \lim_{t\to\infty}\int_{w=0}^{\infty} F_{ij}(t,w)\,dG(w) = \lim_{s\to 0}\int_{w=0}^{\infty} \widetilde{F_{ij}}(s,w)\,dG(w) \\ &= \int_{w=0}^{\infty}\left\{\mathrm{LT}^{-1}\left[(v\mathbf{R}-\mathbf{A})^{-1}\right]\begin{bmatrix}\mathbf{R}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\right\}_{ij} dG(w) \end{aligned} \tag{23}$$

From (23) and (21) the theorem comes. □

From Theorem 4, $P_{ij}^{c} = 0$ if $j \in S^c$. It is the consequence of that the accumulated reward does not increase in S^c and the completion can not occur while $Z(t) \in S^c$.

4 Numerical Example

The results of this paper are demonstrated by the analysis of a simple multiprocessor system. The system is similar to the Carnegie-Mellon multiprocessor system, presented in [15]. The system consists of N processors, M memories, and an interconnection network (i.e., a crossbar switch) that allows any processor to access any memory (Figure 1). The failure rates per hour for the system are set to be 0.2, 0.1 and 0.05 for the processors, memories and the switch, respectively.

Viewing the interconnecting network as one switch and modeling the system at the processor-memory-switch level, the switch becomes essential for the system operation. It is also clear that a minimum number of processors and memories are necessary for the system to be operational. Each state is thus specified by a triple (i, j, k) indicating the number of operating processors, memories, and networks, respectively. We augment the states with the nonoperational state F. Events that decrease the number of operational units are the failures and events that increase the number of operational elements are the repairs. We assume that failures do not occur when the system is not operational. When a component fails, a recovery action must be taken (e.g., shutting down the a failed processor, etc.), or the whole system will fail and enter state F. The probability that the recovery action is successfully completed is known as *coverage*.

Two kinds of repair actions are considered, global repair which restores the system to state $(N, M, 1)$ with rate $\mu = 0.2$ per hour from state F, and local repair, which can be thought of as a repair person beginning to fix a component of the system as soon as a component failure occurs. We assume that there is only one repair person for each component type. Let the local repair rates be 2.0, 1.0 and 0.5 for the processors, memories and the switch, respectively.

The studied system has two processors, two memories, and one connections network, thus the state space consists of 13 states. For this case, the minimal configuration is supposed to have one processor, one memory and one interconnection switch. The value of the coverage was set to 0.90. This is a simple system, however

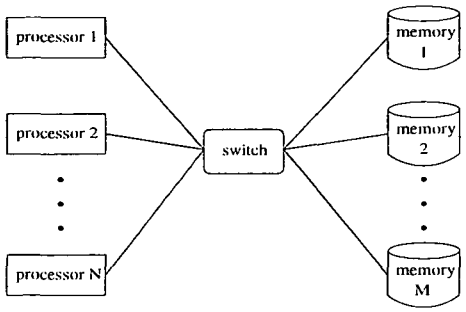


Figure 1: Example system structure

a system of this size would be untractable using the double transformation method. We emphasize that it is just a demonstrative example, the performance of larger systems can also be calculated using the proposed method. More work has to be done to learn the limitations of the proposed method.

The mean value and the standard deviation of the completion time were calculated, the former using Theorem 2, the latter using Theorem 3 and the well known formula $\sigma(w) = (M_{(2)}(w) - (M_{(1)}(w))^2)^{1/2}$. The work requirement was chosen to take values from the interval $[1, 16]$ (in work hours). In Figures 2, 3 the mean value and the standard deviation of the completion time are shown, assuming that the system was started from the perfect state $(N, M, 1)$, from state F and from the steady state distribution. The integral values were calculated numerically in an iterative way. In each step twice as many sample points were evaluated, and the process was stopped when the maximal relative change of the values was less than 2%.

The mean completion time is higher if the system is started in the steady state instead of the perfect $(N, M, 1)$ state, or if the system is started in the F state instead of the steady state. The difference between the perfect and the F initial state curves refers to the mean time to get from state F to the perfect state. The curves of the standard deviation of the completion time show a similar picture. We have to note that the 2% accuracy limit brings more inaccuracy for higher values (8,16). The curve referring to the F state at time 0 takes the value of the standard deviation of the time to get from state F to the perfect state.

5 Conclusion

*MRM*s have been widely used to model performance and reliability of computer and communication systems. We discussed the analytical description of *MRM*s, allowing the assignment of 0 reward rates. A numerically effective computation method is described for the evaluation of the moments of the completion time of a *MRM*. Performance parameters of a Carnegie-Mellon multiprocessor system are evaluated by the proposed method as an application example.

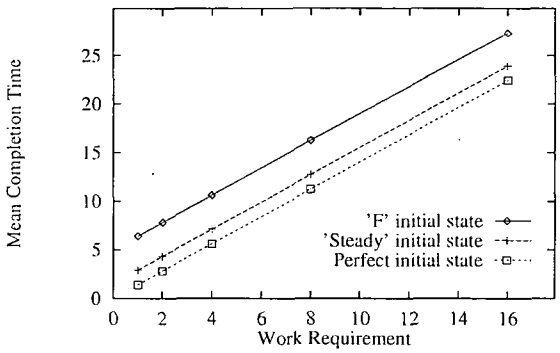


Figure 2: The mean value of the completion time

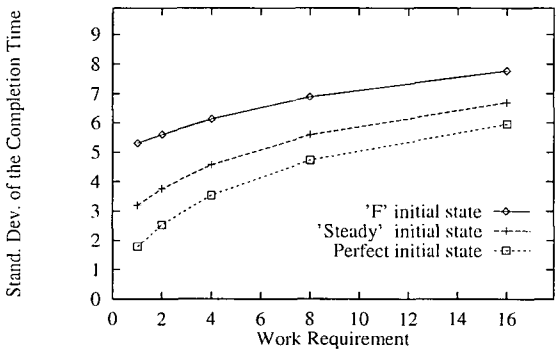


Figure 3: The standard deviation of the completion time

References

[1] M.D. Beaudry. Performance-related reliability measures for computing systems. *IEEE Transactions on Computers*, C-27:540–547, 1978.

[2] K. Begain, L. Jereb, A. Puliafito, and M. Telek. On-off markov reward models. In *Proceedings of Relectronic'95*, pages 95–100, Budapest, 1995.

[3] R. Bellman. *Introduction to matrix analysis.* McGraw Hill, New York, second edition, 1970.

[4] A. Bobbio and L. Roberti. Distribution of the minimal completion time of parallel tasks in multi-reward semi-Markov models. *Performance Evaluation*, 14:239–256, 1992.

[5] A. Bobbio and K.S. Trivedi. Computation of the distribution of the completion time when the work requirement is a PH random variable. *Stochastic Models*, 6:133–149, 1990.

[6] E. Cinlar. Markov renewal theory. *Advances in Applied Probability*, 1:123–187, 1969.

[7] L. Donatiello and V. Grassi. On evaluating the cumulative performance distribution of fault-tolerant computer systems. *IEEE Transactions on Computers*, 1991.

[8] E. De Souza e Silva and H.R. Gail. Calculating availability and performability measures of repairable computer systems using randomization. *Journal of the ACM*, 36:171–193, 1989.

[9] R.A. Howard. *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes.* John Wiley and Sons, New York, 1971.

[10] B.R. Iyer, L. Donatiello, and P. Heidelberger. Analysis of performability for stochastic models of fault-tolerant systems. *IEEE Transactions on Computers*, C-35:902–907, 1986.

[11] V.G. Kulkarni, V.F. Nicola, and K. Trivedi. On modeling the performance and reliability of multi-mode computer systems. *The Journal of Systems and Software*, 6:175–183, 1986.

[12] J.F. Meyer. On evaluating the performability of degradable systems. *IEEE Transactions on Computers*, C-29:720–731, 1980.

[13] A. Pfening, K. Begain, and M. Telek. An effective approach to the completion time analysis on-off Markov reward models. In *European Simulation Multiconference*, pages 796–800, Budapest, June 1996.

[14] Rózsa P. Lineáris algebra és alkalmazásai. *Tankönyvkiadó*, Budapest, 1991.

[15] R. Smith, K. Trivedi, and A.V. Ramesh. Performability analysis: Measures, an algorithm and a case study. *IEEE Transactions on Computers*, C-37:406–417, 1988.

[16] J. Sztrik. Modelling of a multiprocessor system in a randomly changing environment. *Performance Evaluation*, 17:1–11, 1993.

[17] Miklós Telek. *Some advanced reliability modelling techniques.* Phd Thesis, Hungarian Academy of Science, 1994.

Received March, 1998

CONTENTS

Paolo Bottoni, Giancarlo Mauri, Piero Mussio, Gheorghe Păun: Grammars Working on Layered Strings 339
Judit Csima: On extended simple eco-grammar systems 359
Alexander Meduna: Descriptional Complexity of Multi-Continuous Grammars 375
T. Petković, M. Ćirić and S. Bogdanović: Decompositions of automata and transition semigroups 385
László Szabó: On minimal and maximal clones II 405
Csallnér A. E.: Improving Storage Handling of Interval Methods for Global Optimization 413
Günter Rote, Gerhard J. Woeginger: Minimizing the number of tardy jobs .. 423
Attila Sali Sr., Attila Sali: Generalized Dependencies in Relational Databases 431
Miklós Telek, András Pfening, Gábor Fodor: Analysis of the Completion Time of Markov Reward Models 439

Sponsored by SYSDATA Ltd.

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János
A kézirat a nyomdába érkezett: 1998. december
Terjedelem: 7,25 (B/5) ív