Volume 12 Number 2

ACTA CYBERNETICA

Szeged, 1995

Acta Cybernetica 12 (1995) 111-124.

On graphs with perfect internal matchings*

Miklós Bartha† Éva Gombás†

Abstract

Graphs with perfect internal matchings are studied as underlying objects of certain molecular switching devices called soliton automata. A perfect internal matching of a graph is a matching that covers all vertices of the graph, except possibly those with degree one. Such a matching is called a state of the graph. It is proved that for every two states there exists a so called mediator alternating network which can be used as a switch between those two states. As a consequence of this result it is shown how transitions of soliton automata can be decomposed into a sequence of simpler moves. Elementary graphs having a perfect internal matching are defined through an equivalence relation on their edges. Another equivalence relation on the set of vertices is introduced to characterize the well-known canonical partition of elementary graphs in the new generalized sense.

1 Introduction

The results of this paper were motivated by the developments of a research aiming to construct a computer based on molecular switching components. Molecules exhibiting a switching behavior have long been investigated by chemists, cf. [4], but it was not until recently that the first mathematical model of a switching molecular device was introduced in [5] under the name soliton automaton.

The underlying object of a soliton automaton is a so called soliton graph, which is a finite undirected graph modeling the topological structure of a molecule. Atoms are represented by vertices and chemical bonds by edges. The multiplicity of bonds (single or double) is set by a weight assignment to the edges of the corresponding soliton graph. It is assumed that the molecule consists of carbon and hydrogen atoms only, and that among the neighbors of each carbon atom there exists a unique one to which the atom is connected by a double bond. This latter property can nicely be captured by the concept of matching in graphs.

*This work was supported by Natural Science and Engineering Research Council of Canada, Operating Grant 335591 and by National Scientific Research Fund of Hungary, Grant T 014202.

†On leave from Department of Computer Science, József Attila University, Szeged, 6720, Hungary, Department of Computer Science, Memorial University of Newfoundland, St. John's, Newfoundland, Canada A1B 3X5

The above topological model of molecules has already been used earlier to study some other properties of a chemical compound. The reader is referred to [10, Section 8.7] for a detailed discussion on Hückel graphs, which are models of molecules having the same alternating pattern of single and double bonds that we are concerned with in this paper. The only essential difference between Hückel graphs and soliton graphs in terms of matching theory is the following. Hückel graphs are generally supposed to have a perfect matching, whereas in soliton graphs only the internal vertices (i.e. those with degree greater than one) are required to be covered by an appropriate matching. Such a matching is called a perfect internal matching. Vertices with degree one are considered to be external in soliton graphs. The collection of such vertices is treated as an interface for the internal part of the graph, so that these vertices need not be covered by a perfect internal matching.

A state of a soliton automaton is a perfect internal matching of the underlying soliton graph. State transitions are induced by directing a particle (electron, soliton) from one external vertex of the graph to another or even the same external vertex along an alternating walk. Making the walk will then result in a new state by switching all the bonds to the opposite throughout the walk in a dynamic way. For more details, see e.g. [5].

The aim of this paper is to develop a suitable mathematical arsenal for the study of soliton automata. There has already been some previous work done towards this goal. Soliton automata with some special properties have been investigated in [6], [7] and [8]. In [1], an algebraic framework has been introduced to provide a new calculus for dealing with finite undirected multigraphs. Concerning matchings, the Gallai-Edmonds Structure Theorem has been proved for maximum internal matchings in [2]. This theorem plays a central role in the algebra of graphs having a perfect internal matching, which has been described in [3]. Although the present paper is self-contained, some familiarity with [10] will be helpful for the reader.

2 Review of basic concepts and notations

By a graph we mean, throughout the paper, a finite undirected non-empty graph with loops and multiple edges allowed. If G is a graph, then $V(G)$ and $E(G)$ will denote the set of vertices and the set of edges of G, respectively. An edge $e \in E(G)$ connects two vertices $v_1, v_2 \in V(G)$, which are called the *endpoints* of e, and e is said to be *incident* with v_1 and v_2. If $v_1 = v_2$, then e is called a loop around v_1. Two edges sharing at least one endpoint are said to be *adjacent* in G.

For a vertex v in graph G, we define the *degree* of v to be the number of occurrences of v as an endpoint of some edge in $E(G)$. By this definition, the endpoints of a loop are considered to be two different occurrences of the same vertex. The vertex v is called *isolated* if its degree $d(v)$ is zero, *external* if $d(v) = 1$ and *internal* if $d(v) \geq 2$. An edge $e \in E(G)$ is an *internal edge* if both endpoints of e are internal. *External edges* are those that are incident with at least one external vertex.

A *matching* of/in graph G is a subset $M \subseteq E(G)$ such that no vertex of G

occurs more than once as an endpoint of some edge in M. Again, it is understood that loops, having themselves two occurrences of the same endpoint, cannot be present in M. The endpoints of the edges contained in matching M are said to be *covered* by M. A matching M is called *perfect* if it covers all of $V(G)$. A *perfect internal matching* is one that covers all the internal vertices of G. An edge $e \in E(G)$ is *allowed* (*mandatory*) if e is contained in some (respectively, all) perfect internal matching(s) of G. *Forbidden* edges are those that are not allowed. A perfect internal matching in G will also be referred to as a *state* of G.

In graph G, a *trail* is a sequence $\alpha = e_1, \ldots, e_n$ ($n \geq 0$) of distinct adjacent edges $e_i \in E(G)$, $i \in [n] = \{1, \ldots, n\}$ such that no vertex of G occurs more than twice as an endpoint of some e_i. The integer n is said to be the *length* of the trail α. If, in addition, e_n is adjacent to e_1, or $n = 0$, then α is called a *cycle*, otherwise α is a *path*. In the latter case, if e_1 and e_n are both external edges, then α is said to be a *crossing*. Note that every trail α in G can be uniquely specified as an appropriate connected subgraph of G if we are not concerned about the way α is actually traversed. Moreover, if α is non-empty, then this subgraph can be uniquely identified with the set of edges contained in α. Two trails are said to be *disjoint* if they are such as subgraphs of G. Since in this paper, except for Section 5, we shall not be interested in the traversal of trails, we shall often refer to a non-empty trail $\alpha = e_1, \ldots, e_n$ as a set, i.e. as $\alpha = \{e_1, \ldots, e_n\}$, without causing any confusion. Note, however, that disjointness of two trails is ambiguous in general under this assumption.

Let M be a perfect internal matching of G. A trail $\alpha = e_1, \ldots, e_n$ is an *alternating trail* with respect to M (M-alternating trail, for short) if for every $i \in [n-1]$, $e_i \in M$ iff $e_{i+1} \notin M$. An M-alternating trail α is called *complete* if α is either a crossing or it is a non-empty even length cycle. An *alternating network* with respect to M (or M-alternating network) is a set of pairwise disjoint, complete M-alternating trails. Observe that if two complete alternating trails are running on disjoint sets of edges, then they must be disjoint as subgraphs of G, too. Thus, identifying complete alternating trails with the set of their edges does not cause ambiguity regarding the disjointness of such trails. Also note that, although an M-alternating network Γ consists of non-empty trails only, the network Γ itself can be empty.

Let M be a state (i.e., a perfect internal matching) of graph G and α be a complete alternating trail. By *making* α in state M we mean exchanging the status of the edges in α regarding their being present or not present in M. It is easy to see that this process results in another perfect internal matching of G, which will be denoted by $S_G(M, \alpha)$ or simply by $S(M, \alpha)$ if G is understood. Making an M-alternating network Γ in state M means making all the trails of Γ simultaneously in M. Since the trails contained in Γ do not intersect each other, the resulting state, denoted $S_G(M, \Gamma)$, is well-defined.

3 Characterizing state transitions by alternating networks

Our starting theorem relates two arbitrary states of a graph by means of a suitable alternating network that takes the one state into the other. This theorem also manifests the basic inductive proof technique applied in the paper: to obtain a simpler graph, cut an internal edge of the graph at hand and make a correspondence between the complete alternating trails of the original graph and those of the cut graph. Thus, the induction eventually goes by the number of internal edges.

Theorem 3.1 *For any two states M_1, M_2 of graph G, there exists a unique M_1-alternating network Γ for which $S_G(M_1, \Gamma) = M_2$.*

Proof. We prove the existence of Γ by induction on the number of internal edges of G. If G has no internal edges, then all of its components are either star graphs or single edges connecting two external vertices. In such components we can switch from one state to another by making a straightforward crossing.

Suppose now that G has at least one internal edge e, and assume that the assertion holds true for all graphs having fewer internal edges than G. Let v_1 and v_2 denote the two endpoints of e. We cut e by replacing it with two new external edges e_1 and e_2 that are incident with v_1 and v_2, respectively. Let G' denote the resulting graph. Obviously, G' has fewer internal edges than G and it has a perfect internal matching. Moreover, the perfect internal matchings of G are in a one-to-one correspondence with those perfect internal matchings M' of G' for which $e_1 \in M'$ iff $e_2 \in M'$. Applying the induction hypothesis for graph G' and states M_1', M_2' corresponding to the states M_1, M_2 of G, we obtain an M_1'-alternating network $\Gamma' = \{\alpha_1', \ldots, \alpha_k'\}$ in G' satisfying $S_{G'}(M_1', \Gamma') = M_2'$. If neither e_1 nor e_2 is present in any of the trails of Γ', then by putting $\Gamma = \Gamma'$ we are through. Otherwise one of the two cases below is met.

Case 1. There exists a unique $j \in [k]$ such that α_j' is a crossing which connects e_1 to e_2. See Fig. 1a. Since M_1' corresponds to state M_1 of G, we have $e_1 \in M_1'$ iff $e_2 \in M_1'$. This implies that the length of α_j' is odd. Remerging e_1 with e_2 then gives rise to an alternating cycle α_j in G with respect to M_1. Moreover, making α_j' in G' and remerging e_1 with e_2 after has the same effect as making α_j directly in G. Making the network $\Gamma = \Gamma' - \alpha_j' \cup \alpha_j$ in G will therefore take state M_1 to state M_2 as required.

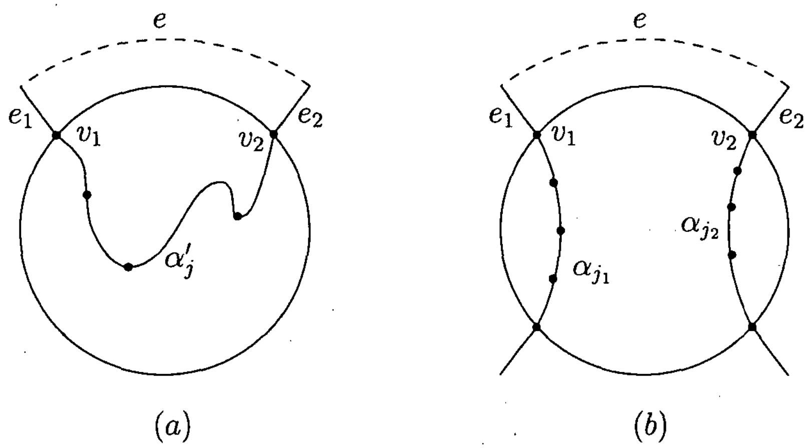


Figure 1.

Case 2. There exist two different crossings α'_{j_i}, $i = 1, 2$, $j_i \in [k]$ such that $e_i \in \alpha'_{j_i}$. See Fig. 1b. In this case the remerging of e_1 and e_2 results in an M_1-alternating crossing α in G. Using the same argument as in Case 1, the desired alternating network is obtained as $\Gamma = \Gamma' - \{\alpha'_{j_1}, \alpha'_{j_2}\} \cup \alpha$.

To prove the uniqueness of Γ we need the following lemma.

Lemma 3.2 *For a graph G, let $\mathcal{C} = \{C_1, \ldots, C_n\}$ and $\mathcal{D} = \{D_1, \ldots, D_m\}$ be two sets of pairwise disjoint connected subgraphs of G. If $\cup\mathcal{C} = \cup\mathcal{D}$, then $\mathcal{C} = \mathcal{D}$.*

Proof. By symmetry it is sufficient to prove that for every $j \in [m]$ there exists some $i \in [n]$ such that $D_j = C_i$. Since the subgraphs contained in $\mathcal{C}$ are pairwise disjoint and D_j is connected, D_j is a subgraph of some C_i, $i \in [n]$. But then D_j must be equal to C_i, otherwise C_i would be covered by more than one subgraph taken from $\mathcal{D}$, contradicting the fact that C_i is connected. □

Now we turn back to the proof of Theorem 3.1. Let us assume that Γ and Δ are complete alternating trails such that $S(M_1, \Gamma) = S(M_1, \Delta) = M_2$. Obviously, both $\cup\Gamma$ and $\cup\Delta$ consist of exactly those edges $e \in E(G)$ for which $e \in M_1$ iff $e \notin M_2$. Therefore $\cup\Gamma = \cup\Delta$, and by Lemma 3.2, $\Gamma = \Delta$.

□

Observe that Theorem 3.1 is symmetric in M_1 and M_2, for Γ is an alternating network with respect to M_1 iff it is one with respect to $M_2 = S_G(M_1, \Gamma)$. In other words,

$$M_1 = S_G(S_G(M_1, \Gamma), \Gamma).$$

The network Γ is called the *mediator* alternating network between states M_1 and M_2, and is denoted by $\mathrm{Med}(M_1, M_2)$.

Let us fix a graph G having a perfect internal matching for Sections 3 and 4. An edge $e \in E(G)$ is said to be *constant* in state M of G if no complete M-alternating trail passes through e.

Corollary 3.3 *An edge e is constant in some state of G iff e is either forbidden* *or mandatory.*

Proof. By Theorem 3.1, e is constant in some state of G iff it is such in all states of G. □

Now we recall the concept of impervious edge from [5]. Although our definition is different from [5, Definition 4.2], it is easy to see that the concepts captured by the two definitions are essentially the same.

Definition 3.4 An edge $e \in E(G)$ is *viable* in state M if there exists an M-alternating path $e_1, \ldots, e_n$ from some external vertex of G to one of the endpoints of e such that

(i) $e \neq e_i$ for any $i \in [n]$;

(ii) e_n and e are M-alternating in the sense that $e_n \in M$ iff $e \notin M$.

The edge e is *impervious* if it is not viable (in state M).

Intuitively, e is viable in state M if there exists an M-alternating path that starts from an external vertex, reaches one endpoint of e without passing through e itself, and it can be continued on e in an alternating fashion. This continuation, however, need not be a path as shown by Fig. 2. In Fig. 2, double lines indicate edges belonging to the matching M rather than multiple edges in the graph G. The reader is referred to [5, Figure 11] for examples of impervious edges in graphs that are connected and have external vertices, too.

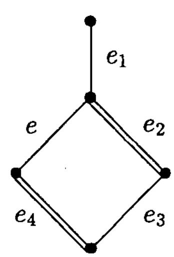


Figure 2.

Corollary 3.5 *An edge is impervious in some state of G iff it is impervious in all states of G.*

Proof. It is sufficient to prove that if $e \in E(G)$ is viable in some state M_1, then it is viable in any other state M_2. Assuming that e is viable in state M_1, let us cut e as described in the proof of Theorem 3.1 to obtain a graph G' with two new external edges e_1, e_2. Again, let M_1' and M_2' be the states of G' corresponding to M_1 and M_2, respectively. By assumption, there exists an M_1'-alternating crossing α' in G' containing exactly one of e_1 and e_2. It follows that exactly one of e_1 and e_2 will be present in some crossing of $\mathrm{Med}(M_2', S_{G'}(M_1', \alpha'))$. From this crossing, after remerging e_1 with e_2, we obtain a suitable M_2-alternating path in G that reaches one endpoint of e and can be continued on e in an alternating fashion. □

4 Elementary equivalence and canonical partition of elementary graphs

Recall from [10] that a graph G is elementary if it has a perfect matching and its allowed edges form a connected subgraph. If G has only a perfect internal matching, then consider the equivalence relation ϵ on $E(G)$ by which $e_1 \,\epsilon\, e_2$ iff either $e_1 = e_2$ or e_1 and e_2 are in the same connected component of the restriction of G to its allowed edges. Our aim is to characterize the relation ϵ in terms of complete alternating trails.

Definition 4.1 Two complete alternating trails α and β with respect to the same state M of G are *conjugated* if $\mathrm{Med}(S(M,\alpha), S(M,\beta))$ is a singleton.

It is immediate by the above definition that if α and β are conjugated, then they must intersect each other without being identical themselves. Indeed, if α and β are complete alternating trails, then $\alpha \cap \beta = \emptyset$ implies that $\mathrm{Med}(S(M,\alpha), S(M,\beta)) = \{\alpha, \beta\}$ and $\alpha = \beta$ implies that $\mathrm{Med}(S(M,\alpha), S(M,\beta)) = \emptyset$. (Remember that all complete alternating trails are non-empty, by definition.)

Lemma 4.2 *Let α and β be two complete alternating trails with respect to the same state M. Then, for every edge $e \in \beta - \alpha$ there exists a complete alternating trail γ with respect to some $\hat{M} \in \{M, S(M,\alpha)\}$ passing through e such that α and γ are either conjugated or disjoint.*

Proof. Let $n_\beta(\alpha)$ be the number of edges contained in $\beta - \alpha$. The proof is an inductive argument on $n_\beta(\alpha)$. The basis case $n_\beta(\alpha) = 0$ is trivial.

Let $n_\beta(\alpha) \geq 1$, and assume that the assertion of the lemma holds for all triples (α', β', M') such that α' and β' are complete alternating trails with respect to state M', and $n_{\beta'}(\alpha') < n_\beta(\alpha)$. Choose β' to be the complete alternating trail of the network $\Gamma = \mathrm{Med}(S(M,\alpha), S(M,\beta))$ containing e. Since the trails of Γ are running exclusively on those edges of α and β that are not contained in their intersection, $n_{\beta'}(\alpha) \leq n_\beta(\alpha)$. Moreover, $n_{\beta'}(\alpha) = n_\beta(\alpha)$ iff either $\beta' = \beta$ is disjoint from α or Γ is a singleton, in which cases there is nothing to prove. If, however, $n_{\beta'}(\alpha) < n_\beta(\alpha)$, then the induction hypothesis can be applied for $\alpha' = \alpha$ and β', which are complete alternating trails with respect to state $M' = S(M,\alpha)$. To complete the proof, one must take into account that $S(S(M,\alpha),\alpha) = M$.

□

Corollary 4.3 *Let α be a complete alternating trail in G with respect to state M, and let $e \in E(G)$ be an allowed edge adjacent to some edge in α. Then for every $e' \in \alpha$ there exists a complete alternating trail δ with respect to some $\hat{M} \in \{M, S(M,\alpha)\}$ which contains both e and e'.*

Proof. We can assume that $e \notin \alpha$. If e were mandatory, then every edge adjacent to e would be forbidden, contradicting the fact that all the edges of α are allowed. Thus, by Corollary 3.2, there exists a complete alternating trail β with $e \in \beta - \alpha$. Applying Lemma 4.2 we obtain a complete alternating trail γ with respect to some $\hat{M} \in \{M, S(M,\alpha)\}$ which also contains e and, moreover, α and γ are conjugated.

It is now clear that the required complete alternating trail δ can be chosen either as $\delta = \gamma$ or as $\delta = \mathrm{Med}(S(\hat{M},\alpha), S(\hat{M},\gamma))$, depending on whether γ passes through e or not.

□

Now we redefine the relation of elementary equivalence introduced at the beginning of this section.

Definition 4.4 Two edges e_1, e_2 are *elementary equivalent* if either $e_1 = e_2$ or there exists a complete alternating trail with respect to some state of G containing both e_1 and e_2.

The relation of elementary equivalence will be denoted by ϵ.

Theorem 4.5 *Elementary equivalence is an equivalence relation on $E(G)$.*

Proof. We only have to address transitivity of ϵ. Let e_1, e_2, e_3 be such that $e_1 \,\epsilon\, e_2$ and $e_2 \,\epsilon\, e_3$. Then there exists a complete M_1-alternating trail α joining e_1 to e_2 and a complete M_2-alternating trail joining e_2 to e_3, where M_1 and M_2 are some states of G. It follows that e_3 can be reached from some vertex lying on α by a path τ consisting of allowable edges only. Using Corollary 4.3, a straightforward induction on the length of τ shows that there exists a complete alternating trail δ with respect to some state M_3 which contains both e_1 and e_3. Thus, $e_1 \,\epsilon\, e_3$, which was to be proved.

□

It turns out from the proof of Theorem 4.5 that if e_1 and e_2 are adjacent edges in G, then either $e_1 \,\epsilon\, e_2$ or one of e_1 and e_2 is forbidden. This means that the relation ϵ coincides with the one that we intended to characterize at the beginning of this section. Spelling this out, the equivalence classes of ϵ that are different from a single forbidden edge are exactly the connected components of the restriction of G to its allowed edges.

Consider the relation ϵ_V on $V(G)$ by which $v_1 \,\epsilon_V\, v_2$ iff either $v_1 = v_2$ or v_1 and v_2 can be connected by a complete alternating trail with respect to some state of G. By virtue of Theorem 4.5, ϵ_V is also an equivalence relation. Slightly modifying the original definition given in [10], we call G *elementary* if ϵ_V is the universal relation on $V(G)$. Note that if G is elementary, then the relation ϵ is not necessarily universal on $E(G)$, for G might contain some forbidden edges as well.

For the rest of this section we shall assume that G is elementary. Our goal is to find the analog of the canonical partition $\mathcal{P}(G)$ of $V(G)$, where G is a graph with a perfect matching, for the case when G has just a perfect internal matching. The partition $\mathcal{P}(G)$ has been described in [10, Theorem 5.2.2] in many different ways, based on the concepts of extreme set and barrier. Unfortunately, we have not been able to generalize these concepts for graphs with perfect internal matchings yet, but we can still characterize $\mathcal{P}(G)$ by the following very simple relation $\sim$.

Definition 4.6 For two vertices $v_1, v_2 \in V(G)$, $v_1 \sim v_2$ if an extra edge e connecting v_1 and v_2 becomes forbidden in $G + e$ (i.e. in the extension of G by e).

According to part (b) of [10, Theorem 5.2.2], if G does not contain loops and external vertices, then $\sim$ is an equivalence relation and $\mathcal{P}(G)$ is the partition induced by $\sim$. Here we prove that $\sim$ is an equivalence anyhow.

Theorem 4.7 *The relation $\sim$ is an equivalence on $V(G)$.*

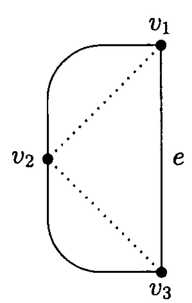
Proof. Since loops are forbidden edges, $\sim$ is reflexive. It remains to show the transitivity of $\sim$. Let $v_1 \sim v_2$ and $v_2 \sim v_3$ for distinct vertices $v_1, v_2, v_3 \in V(G)$. We have to prove that an extra edge e connecting v_1 with v_3 becomes forbidden in $G+e$. Assume, on the contrary, that e is allowed. It is clear that e is not mandatory, hence $G+e$ is still elementary. Moreover, due to the elementary property, there is an allowed edge e' incident with v_2 such that $e \epsilon e'$ holds in $G+e$. Therefore, by Theorem 4.5, there exists a complete alternating trail α with respect to some state M of $G+e$ containing e and reaching v_2 on the way. We distinguish two cases.

Case 1: α is an even length cycle, see Fig. 3a.

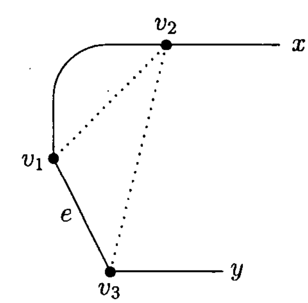
Since each of the edges (v_1, v_2) and (v_2, v_3) would become forbidden when adding them to G, the subpaths of α connecting v_1 with v_2 and v_2 with v_3 are of even length. Thus, together with e, the length of α turns odd, which is a contradiction.

Case 2: α is a crossing that connects two external vertices x, y, see Fig. 3b.

Without loss of generality we may assume that v_1 lies between v_2 and v_3 on α and that $e \notin M$. Again, the length of the subpath of α connecting v_1 and v_2 is even, for $v_1 \sim v_2$.



(a)



(b)

Figure 3.

Consequently, the crossing $x, \ldots, v_2, v_3, \ldots, y$ is M-alternating in $G + (v_2, v_3)$, contradicting the assumption that $v_2 \sim v_3$.

□

5 Connection to soliton automata

According to [5], a soliton graph is a pair (G, w), where G is an undirected graph and w is a weight function from $E(G)$ into the set of positive integers such that these data satisfy the following conditions:

(a) G has no loops or multiple edges;

(b) every connected component of G has at least one external node;

(c) for every $v \in V(G)$, $d(v) \leq 3$;

(d) for every internal vertex v, $w(v) = d(v) + 1$, where $w(v)$ stands for the sum of the weights of all edges incident with v;

(e) if v is an external vertex, then $w(v) \in \{1, 2\}$.

Conditions (d) and (e) imply that the weight of every edge in G is either 1 or 2, and for every internal vertex v there exists exactly one edge e incident with v such that $w(e) = 2$. Let $M \subseteq E(G)$ consist of those edges which have weight 2. Clearly, M is a perfect internal matching of G. Conversely, every perfect internal matching of G corresponds to a suitable weight function w satisfying (d) and (e) above. Hence our approach to soliton automata based on matching theory. Conditions (a), (b) and (c) impose restrictions on the graph structure only, so that they are irrelevant as far as matchings are concerned. We believe that the concept "soliton graph" should be independent of the particular weight function (perfect internal matching) chosen for it, that is why we would rather define a soliton graph simply to be a graph having a perfect internal matching.

Now we quote the definition of soliton path from [5]. Note that the terminology of the authors of [5] differs from ours in that they call a path what we defined as a walk in Section 2. Moreover, since their discussion excludes graphs with loops and multiple edges, it was sufficient for them to specify a path as a sequence of vertices rather than a sequence of edges.

Thus, according to [5], a *partial soliton path* in a soliton graph (G, w) is a path $v_0, v_1, \ldots, v_k$ satisfying the following conditions:

(a) v_0 is an external vertex;

(b) $v_1, v_2, \ldots, v_{k-1}$ are internal vertices;

(c) there is a sequence $(G, w_0), \ldots, (G, w_k)$ of weighted (not necessarily soliton) graphs

that are constructed as follows:

(c1) $w_0 = w$;

(c2) for $i = 0, 1, \ldots, k-2$, the function w_{i+1} is defined iff w_i is defined and $|w_i(v_i, v_{i+1}) - w_i(v_{i+1}, v_{i+2})| = 1$. In this case, for all edges $(v, v') \in E(G)$,

$$w_{i+1}(v, v') = \begin{cases} w_i(v, v') & \text{if } (v, v') \neq (v_i, v_{i+1}) \\ 3 - w_i(v_i, v_{i+1}) & \text{if } (v, v') = (v_i, v_{i+1}); \end{cases}$$

(c3) w_k is defined iff w_{k-1} is defined. In this case, for all $(v,v') \in E(G)$,

$$w_k(v,v') = \begin{cases} w_{k-1}(v,v') & \text{if } (v,v') \neq (v_{k-1},v_k) \\ 3 - w_{k-1}(v_{k-1},v_k) & \text{if } (v,v') = (v_{k-1},v_k). \end{cases}$$

A partial soliton path is called a *(total) soliton path* if v_k above is an external vertex.

Intuitively, a soliton path (walk) is an alternating walk with respect to some state M of the graph G that starts and ends at an external vertex. However, the status of each edge in the walk regarding its presence in M changes dynamically step by step while making the walk. More precisely, this status changes to the opposite right after having traversed the edge. Thus, by the time the walk is finished, a new state M' of G is reached. See [5, Lemma 3.3] for a proof of this last statement.

Here we provide a somewhat simpler definition of soliton walks using our own terminology. For the sake of convenience and unambiguity, we shall specify a walk of length n in graph G as a sequence $\alpha = v_0, e_1, \ldots, e_n, v_n$ of alternating vertices and edges, indicating also the starting point $v_0 \in V(G)$ of α and the vertex v_j, $j \in [n]$, that the walk has reached after traversing the j-th edge e_j. For every $j \in [n]$, $n_\alpha(j)$ will denote the number of occurrences of the edge e_j in the prefix $v_0, e_1, \ldots, e_j$. By a *backtrack* in a walk we mean the traversal of the same edge twice in a consecutive way.

Let us again fix a graph G having a perfect internal matching for the rest of this section.

Definition 5.1 A *soliton walk* in G with respect to state M is a walk $\alpha = v_0, e_1, \ldots, e_n, v_n$ subject to the following two conditions:

(a) v_0 and v_n are external vertices with $n \geq 1$;

(b) for every $j \in [n-1]$, $n_\alpha(j)$ and $n_\alpha(j+1)$ have the same parity iff e_j and e_{j+1} are

M-alternating, i.e., $e_j \in M$ iff $e_{j+1} \notin M$.

It is left to the reader to check that, for soliton graphs in the sense of [5], Definition 5.1 is equivalent to the above definition of soliton path with the only difference that we allow soliton walks to make a backtrack on external edges, too. Any backtrack in a soliton walk is, however, a redundant move as shown by Proposition 5.2 below. *Making* the walk α in state M means creating a new state $M' = S(M,\alpha)$ by setting for every $e \in E(G)$

$$e \in M' \text{ iff } \begin{cases} e \in M & \text{and } e \text{ occurs an even number of times in } \alpha \\ \text{or} & \\ e \notin M & \text{and } e \text{ occurs an odd number of times in } \alpha. \end{cases}$$

In the light of [5, Lemma 3.3] it should be clear that $S(M,\alpha)$ is indeed a state.

Proposition 5.2 *For every soliton walk α with respect to some state M there exists a backtrack-free soliton walk β with respect to M such that $S(M,\alpha) = S(M,\beta)$.*

Proof. Obvious induction on the number of backtracks contained in α, omitted. □

Now we reformalize the definition of soliton automata [5] in our matching theoretic framework.

Definition 5.3 A *soliton automaton* with underlying graph G is a finite state nondeterministic automaton

$$\mathcal{A}(G) = (S(G), X \times X, \delta)$$

subject to the following conditions:

(a) G has a perfect internal matching and has at least one external vertex;

(b) $S(G)$, the set of states of $\mathcal{A}(G)$, is the set of all states of G;

(c) $X \times X$ is the input alphabet, where $X \subseteq V(G)$ denotes the subset of all external vertices.

(d) $\delta : S(G) \times (X \times X) \to 2^{S(G)}$ is the transition function defined as follows. For every state M and external vertices $v_1, v_2 \in X(G)$, $M' \in \delta(M, (v_1, v_2))$ if there exists a soliton walk α from v_1 to v_2 with respect to M such that $M' = S(M, \alpha)$.

Let α be a soliton walk in G with respect to state M. From Theorem 3.1 we know that making α is equivalent to making an appropriate alternating network Γ with respect to M. The network Γ will consist of a number of alternating cycles $\beta_1, \ldots, \beta_n$, and, in the case when the two endpoints of α are distinct, an additional crossing γ. We are going to prove that the cycles $\beta_1, \ldots, \beta_n$ can be made separately one after the other as suitable soliton walks from the starting point of α back to the same vertex in such a way that making these walks and then finishing up with γ has the same effect as making α directly in state M. This result will admit a decomposition of the transitions of the automaton $\mathcal{A}(G)$ into simpler ones.

Lemma 5.4 *For any state M of G let Γ be an alternating network consisting of a number of cycles. Furthermore, let v_0 be an external vertex and $v \in V(G)$ be arbitrary. If there is an alternating path from v_0 to v with respect to M, then there is also one with respect to $S(M, \Gamma)$.*

Proof. This lemma is in fact a consequence of Corollary 3.4. Let α be an M-alternating path from v_0 to v. Without loss of generality we can assume that α is non-empty, i.e., $v_0 \neq v$. Then the last edge e of α is incident with v and it is viable in state M. Therefore, by Corollary 3.4, e is viable in state $S(M, \Gamma)$, too. Let v' denote the other endpoint of e. By checking the proof of Corollary 3.4 the reader can verify that the alternating path α' demonstrating that e is viable in state $S(M, \Gamma)$ will consist of those edges only that are either in α or in $\cup\Gamma$. Consequently, since $\cup\Gamma$ does not contain any external edges, the path α' will connect v_0 with either

v or v', and it will have an alternating continuation on e. From this point the proof is obvious. □

Let v be an external vertex of G. A soliton walk α is called a *v-saucepan* if it can be decomposed in the form $\alpha\beta\alpha^{-1}$, where α is an alternating path from v to some internal vertex u, β is an alternating cycle starting and ending at u such that β does not go through any vertices covered by α, and α^{-1} is the reverse of α. See Fig. 2 for an example of a saucepan.

Theorem 5.5 *Let $\beta_1, \ldots, \beta_n$ be disjoint alternating cycles with respect to state M that are reachable from an external vertex v_0 of G by a suitable M-alternating path. Then for every $i \in [n]$ there exists a v_0-saucepan δ with respect to state $S(M, \{\beta_1, \ldots, \beta_{i-1}\})$ such that*

$$S(S(M, \{\beta_1, \ldots, \beta_{i-1}\}), \delta_i) = S(M, \{\beta_1, \ldots, \beta_i\}).$$

Proof. Induction on n. The basis case $n = 0$ is vacuously true. Assuming that the statement holds for some $n \geq 0$, let $\beta_1, \ldots, \beta_{n+1}$ be alternating cycles satisfying the conditions of the theorem. By assumption, there exists an M-alternating path α to some vertex v lying on β_{n+1} which starts from v_0 and does not go through any vertices lying on β_{n+1}. Lemma 5.4 then implies that there is another alternating path α' with respect to $S(M, \{\beta_1, \ldots, \beta_n\})$ having the same properties. Therefore we can compose the required v_0-saucepan δ_{i+1} by going down to v from v_0 on α', making the cycle β_{i+1}, and returning to v_0 on the reverse of α'. □

Corollary 5.6 *Every transition of $A(G)$ on input (v_1, v_2) can be decomposed into a sequence of simpler transitions induced by suitable soliton walks $\beta_1, \ldots, \beta_n, \beta_{n+1}$ such that β_i is a v_1-saucepan for every $i \in [n]$, and, in the case of $v_1 \neq v_2$, β_{n+1} is a crossing from v_1 to v_2 .*

Proof. Immediate by Theorems 3.1 and 5.5. □

6 Conclusion

We have made a few simple observations on graphs having a perfect internal matching and on soliton automata. The heart of our results is Theorem 3.1, which specifies the relationship between two perfect internal matchings in a graph in terms of alternating paths and cycles. We have also introduced two equivalence relations ϵ and $\sim$. For a graph G, the relation ϵ can be used to isolate elementary components within the restriction of G to its allowed edges, while the equivalence $\sim$ tells which of the vertices contained in the same elementary component are or could be connected by a forbidden edge in G. Finally, we devoted a section to specify the connection between our work and the study of soliton automata. We would like to use the main result of this section, Theorem 5.5, to provide a decomposition of soliton automata in the spirit of [9].

It is not only the mere fact that we are dealing with perfect internal matchings instead of perfect matchings which makes our results different from the corresponding existing or nonexisting ones in classical matching theory. The difference also appears in the technique by which we prove these results. Rather than using the trick of deleting an appropriate vertex or several vertices in a graph, which seems to be dominant in the classical approach, we almost exclusively rely on the operations of cutting and remerging the edges of graphs. This technique makes our approach edge-oriented as opposed to the vertex-oriented classical approach. Our way of thinking about matchings is based entirely on the method of dealing with alternating paths and cycles, and it fits into the algebraic framework outlined in [1] and [3].

References

[1] M. Bartha, H. Jürgensen, Characterizing finite undirected multigraphs as indexed algebras, Research Report # 252, Department of Computer Science, The University of Western Ontario, London, Canada, 1989.

[2] M. Bartha, E. Gombás, A structure theorem for maximum internal matchings in graphs, *Information Processing Letters* **40** (1991), 289–294.

[3] M. Bartha, E. Gombás, The Gallai-Edmonds algebra of graphs, Research Report # 9105, Department of Computer Science, Memorial University of Newfoundland, St. John's, Canada, 1991.

[4] F. L. Carter, Comformational switching at the molecular level, in *Molecular Electronic Devices* (F. L. Carter ed.), Marcel Dekker, Inc., New York, 1982, pp. 51–72.

[5] J. Dassow, H. Jürgensen, Soliton automata, *J. Comput. System Sci.* **40** (1990), 154–181.

[6] J. Dassow, H. Jürgensen, Soliton automata with a single exterior node, *Theoretical Computer Science* **84** (1991), 281–292.

[7] J. Dassow, H. Jürgensen, Soliton automata with at most one cycle, *Journal of Computer and System Sciences* **46** (1993), 155–197.

[8] J. Dassow, H. Jürgensen, The transition monoids of soliton trees, Research Report # 331, Department of Computer Science, The University of Western Ontario, London, Canada, 1992.

[9] F. Gécseg, H. Jürgensen, Automata represented by products of soliton automata, *Theoretical Computer Science* **74** (1990), 163–181.

[10] L. Lovász, M. D. Plummer, *Matching Theory*, North Holland, Amsterdam, 1986.

Received January, 1995

Acta Cybernetica 12 (1995) 125-129.

Remarks on the Interval Number of Graphs

A. Pluhár *

Abstract

The interval number of a graph G is the least natural number t such that G is the intersection graph of sets, each of which is the union of at most t intervals. Here we propose a family of representations for the graph G, which yield the well-known upper bound $\lceil \frac{1}{2}(d+1) \rceil$, where d is the maximum degree of G. The extremal graphs for even d are also described, and the upper bound on the interval number in terms of the number of edges of G is improved.

1 Introduction and Results

It is a very natural idea to represent a graph G as the intersection graph of some sets. That is, we assign a set to each vertex of G so that v is adjacent to w if and only if the common part of the assigned sets is not empty. A $t-$*interval representation* is an assignment, where each set consists of at most t closed intervals. The *interval number* of G, denoted by $i(G)$, is the least integer t for which a t-representation of G exists. Finally, a representation is *displayed* if each set of the representation has an open interval disjoint from the other sets. Such an interval is called *displayed segment*.
There are a number of published results concerning bounds on $i(G)$, as well as applications of the interval representations [1-8]. Since for the complete graph K_n (on n vertices) $i(G) = 1$, the main interest lies in finding *upper bounds* in terms of the maximum degree, the number of vertices and the number of edges of a graph G, see in [2], [3], [6] and [8].

Theorem 1 (3) *If G is a graph with maximum degree d, then* $i(G) \leq \lceil \frac{1}{2}(d+1) \rceil$.

The bound of Theorem 1 is sharp, since the equality is attained for example a d-regular, triangle-free graphs G. We shall give a new proof of Theorem 1, which is also useful in investigating the extremal graphs of the degree bound.

Theorem 2 *If a graph G has no d-regular, triangle-free component, then* $i(G) \leq \lceil \frac{1}{2}d \rceil$.

*Department of Computer Science, Attila József University, Árpád tér 2, H-6720 Szeged, Hungary

That is to say, in the case $d = 2k$ the extremal graphs are just the d-regular, triangle-free graphs. Unfortunately, one cannot expect to get such a simple result when $d = 2k+1$. For example the graph which arise from $K_{1,3}$ subdividing all its edges [7], or C_n, $n \geq 5$ with a chord have interval number 2 with $d = 3$.
It is possible to bound $i(G)$ in terms of e, where e is the number of edges in G. It was conjectured in [3] that $i(G) \leq \lceil \frac{1}{2}\sqrt{e}\, \rceil + 1$, which would be best possible because of the graphs $K_{2m,2m}$ for $m \in N$. The best published result is in [6], stating that $i(G) \leq \lceil \sqrt{\frac{e}{2}}\, \rceil + 1$. We shall improve on the estimations used in [6], and show

Theorem 3 *Every graph with e edges has a displayed interval representation with at most* $1 + \lceil \frac{2}{3}\sqrt{e}\, \rceil$ *intervals for each vertex.*

It is necessary to state one more earlier result in order to prove Theorem 3.

Theorem 4 (2) *If a graph G has* $n > 1$ *vertices, then* $i(G) \leq \lceil \frac{1}{4}(n+1) \rceil$, *and this bound is the best possible.*

2 Proofs

Proof of Theorem 1

We shall construct a displayed representation for the graph G such that at most $\lceil \frac{1}{2}(d(v)+1) \rceil$ intervals are assigned to each vertex v, where $d(v)$ designates the degree of the vertex v. A *walk* W in G is just a sequence of vertices $W = \{v_1, v_2, ..., v_l\}$ such that, there is an edge between v_i and v_{i+1} for each $i = 1, 2, ..., l-1$. Let us partition the edges of G into minimal number of edge disjoint walks $\{W_i\}_{i=1}^{j}$. Now represent the walk $W_i = (v_1^i, v_2^i, ..., v_{n(i)}^i)$ for $1 \leq i \leq j$, assigning an I_p^i interval to the vertex v_p^i such that two intervals have intersection if and only if the corresponding vertices are next to each other in the walk W_i. This procedure leads to a displayed interval representation of G. Since a vertex v can be an endvertex of the walks at most two times, if v is represented by l intervals, then $d(v) \geq 2(l-2) + 2 = 2l - 2$. Hence

$$\lceil \frac{1}{2}(d(v)+1) \rceil \geq \lceil \frac{1}{2}(2l-2+1) \rceil = \lceil l - \frac{1}{2} \rceil = l.$$

□

Proof of Theorem 2

We can assume that $d = 2k$ because of Theorem 1. Let us choose among all partitions of the edge set into a minimum number of edge disjoint walks a partition $\{W_i\}_{i=1}^{j}$ which also minimizes the size of the set Q of vertices occuring $k+1$ times in the walks $\{W_i\}_{i=1}^{j}$. The representation is the same as in the proof of Theorem 1. If $Q = \emptyset$, we are done. For an $x \in Q$ there exists a $p \in \{1, ..., j\}$ such that $x = v_1^p$, $x = v_{n(p)}^p$ and $x \notin W_l$ for all $l \neq p$. The last statement follows from the minimality of j, since in case of $x = v_s^l \in W_l$ we could replace the walks W_p and W_l by the walk

$$W^* = (v_1^l, v_2^l, ..., v_s^l, v_2^p, ..., v_{n(p)}^p, v_{s+1}^l, ..., v_{n(l)}^l).$$

For any vertex $y = v_s^p \neq x$ from W_p, we can transform the walk W_p into the walk

$$W_p^* = (v_s^p, v_{s-1}^p, ..., v_1^p, v_{n(p)-1}^p, v_{n(p)-2}^p, ..., v_s^p).$$

That is, by the minimality of Q, y occurs in the walks $\{W_i\}_{i \neq p} \cup \{W_p^*\}$ $k+1$ times. Then again, all neighbors of y are in W_p. That is the vertex set of W_p is a $2k-$regular component of G. Now we can conclude the proof by showing that if a $2k-$regular graph G is not triangle-free, then $i(G) \leq k$. Suppose that u, v and w span a triangle in G. If $k = 1$, then $G = K_3$, and we are done. For $k > 1$ there is is an Euler circuit C in G, starting by v, u, w, v, x and finishing at v. But it can be represented by k intervals per vertex as in the proof of Theorem 1, just take the convex hull of the two intervals which represent v at the beginning of the walk. □

Proof of Theorem 3

We need the definition of the *degree sequence* of a graph G first. Let us suppose that $v_1, ..., v_n$ is an order of the vertices of G such that $d_i \geq d_j$ if $i \leq j$, where $d_i = deg(v_i)$ denotes the degree of the vertex v_i. Our argument closely follows the one in [8]. The crucial difference is the additional information about the degree sequence of G. It is gained by using Theorem 1 and an idea, which first appeared in [4].

Lemma 1 *Let $d_1 \geq d_2 \geq ... \geq d_n$ be the degree sequence of a graph G. If $i(G) > t+1$, then $d_i \geq 2t - i + 1$.*

Proof of Lemma 1

Let v_i be a vertex of degree d_i. By Theorem 1

$$\lceil \frac{1}{2}(d_1 + 1) \rceil \geq i(G) > t + 1,$$

that is $d_1 \geq 2t+2$. Now we partition the edges of G into directed forests, represent them one by one and remove the edges of the represented forest from G. The idea is that the representation of the l^{th} forest exhausts all edges adjacent to v_l, and decreases the degree of all vertices in the remaining graph which still has non zero degree. The construction of the first forest F_1 starts with choosing a breadth-first-search tree T_1, rooted in v_1, all edges directed toward v_1. If there are vertices outside of T_1, just pick arbitrary trees in which the edges are directed toward the root. The procedure for selecting F_l is similar, we take v_l, the vertex of degree d_l as a root of a tree, and also take other trees if the remaining graph is not connected. The main point is that F_l is maximal, and all edges adjacent to v_l are in $F_1 \cup ... \cup F_l$. For the maximum degree Δ^i in the remaining graph $G^i = G \backslash F_1 \cup ... \cup F_{i-1}$ we have show that

$$\Delta^i \leq d_i - (i-1)$$

by induction. On the other hand, we can represent the edges of $F_1 \cup ... \cup F_l$ by using at most $l+1$ intervals for each vertex. First assign intervals I_v to each vertex v of G such that $I_v \cap I_w = \emptyset$ for $v \neq w$. Then, for each $i \in \{1, ..., l\}$ if the directed edge (v, w) is in F_i, assign a small interval to v inside in I_w, which has no common points with the other intervals.
Because of Theorem 1 and the previous representation we have

$$i(G) \leq i + i(G \backslash F_1 \cup ... \cup F_{i-1}) \leq i + \lceil \frac{d_i - (i-1) + 1}{2} \rceil.$$

Since $t + 1 < i(G) \leq i + \lceil \frac{d_i - i + 2}{2} \rceil$, it follows that

$$t + 3/2 \leq i + \frac{d_i - i + 2}{2},$$

that is $d_i \geq 2t - i + 1$. □

Now, with a few modifications, we may repeat the argument presented in [8]. First, partition the vertices of G into two classes, A and B. A contains the vertices of degree at least $\lceil \frac{2}{3}\sqrt{e} \rceil + 1$, while the degree of a vertex from B is at most $\lceil \frac{2}{3}\sqrt{e} \rceil$. The edges between the elements of A can be represented by using at most $\lceil \frac{1}{4}(|A| + 1) \rceil$ intervals for each vertex because of Theorem 4. Let us make this system of intervals displayed by adding an isolated interval for each vertex of G in a same way as in the proof of Lemma 1. For each edges between A and B, or inside B, take an endpoint from B, and place a small interval for it into a displayed segment for its neighbor. This procedure produces at most $\lceil \frac{2}{3}\sqrt{e} \rceil + 1$ intervals for an element of B. That is

$$i(G) \leq \max(\lceil \frac{2}{3}\sqrt{e} \rceil + 1, \lceil \frac{1}{4}(|A| + 1) \rceil + 1).$$

In order to estimate $|A| = k$, we need the identity $2e = \sum_{i=1}^{n} d_i$, where $\{d_i\}_{i=1}^{n}$ is the degree sequence in decreasing order. There is nothing to prove if $i(G) \leq \lceil \frac{2}{3}\sqrt{e} \rceil + 1$, so we may assume that

$$d_i \geq 2\lceil \frac{2}{3}\sqrt{e} \rceil - i + 1$$

by Lemma 1. Thus

$$2e = \sum_{i=1}^{\lceil \frac{2}{3}\sqrt{e} \rceil} d_i + \sum_{i=\lceil \frac{2}{3}\sqrt{e} \rceil + 1}^{k} d_i + \sum_{i=k+1}^{n} d_i,$$

which implies

$$2e \geq \sum_{i=1}^{\lceil \frac{2}{3}\sqrt{e} \rceil} (2\lceil \frac{2}{3}\sqrt{e} \rceil - i + 1) + \sum_{i=\lceil \frac{2}{3}\sqrt{e} \rceil + 1}^{k} (\lceil \frac{2}{3}\sqrt{e} \rceil + 1).$$

Simple computation shows that $k \leq \frac{8}{3}\sqrt{e} - 1$. Plugging in this estimation, one gets the bound

$$i(G) \leq \max(\lceil \frac{2}{3}\sqrt{e} \rceil + 1, \lceil \frac{1}{4}(\frac{8}{3}\sqrt{e} - 1 + 1) \rceil + 1) \leq \lceil \frac{2}{3}\sqrt{e} \rceil + 1.$$

□

References

[1] Andreae, T., "On the Interval Number of Triangulated Graph", *J. Graph Theory* **3** (1987), 273-280.

[2] Griggs, J.R., "Extremal Values of the Interval Number of Graph, II.", *Discrete Math.* **28** (1979), 37-47.

[3] Griggs, J.R. and West, D.B., "Extremal Values of the Interval Number of Graph, I.", *SIAM J. Alg. Discrete Meth.* **1** (1980), 1-8.

[4] Scheinerman, E.R., "The Maximum Interval Number of Graphs with Given Genus", *J. Graph Theory* **3** (1987), 441-446.

[5] Scheinerman, E.R. and West, D.B., "The interval number of a planar graph: three intervals suffice", *Jour. Combinatorial Theory B* **35** (1985), 224-239.

[6] Spinrad, J.R., Vijayan, G. and West, D.B., "An Improved Edge Bound on the Interval Number of a Graph", *J. Graph Theory* **3** (1987), 447-449.

[7] Trotter, W.T. Jr. and Harary, F. "On Double and Multiple Interval Graphs", *J. Graph Theory* **3** (1979), 205-211.

[8] West, D.B., "A short proof of the degree bound on interval number", *Discrete Math.* **73** (1989), 307-310.

Received May, 1995

Acta Cybernetica 12 (1995) 131-136.

Right group-type automata*

I. Babcsányi † A. Nagy†

Abstract

In this paper we deal with state-independent automata whose characteristic semigroups are right groups (left cancellative and right simple). These automata are called right group-type automata. We prove that an A-finite automaton is state-independent if and only if it is right group-type. We define the notion of the right zero decomposition of quasi-automata and show that the state-independent automaton $\mathbf{A}$ is right group-type if and only if the quasi-automaton $\mathbf{A}_S^*$ corresponding to $\mathbf{A}$ is a right zero decomposition of pairwise isomorphic group-type quasi-automata. We also prove that the state-independent automaton $\mathbf{A}$ is right group-type if and only if the quasi-automaton $\mathbf{A}_S^*$ corresponding to $\mathbf{A}$ is a direct sum of pairwise isomorphic strongly connected right group-type quasi-automata. We prove that if $\mathbf{A}$ is an A-finite state-independent automaton, then $|S(A)|$ is a divisor of $|AS(A)|$. Finally, we show that the quasi-automaton $\mathbf{A}_S^*$ corresponding to an A-finite state-independent automaton $\mathbf{A}$ is a right zero decomposition of pairwise isomorphic quasi-perfect quasi-automata if and only if $|AS(A)| = |S(A)|$.

In his paper [5], A. C. Fleck introduced the notion of the characteristic semigroup of automata. This notion is a very useful tool for the examination of automata from semigroup theoretical aspects. In particular, it seems to be successful for state-independent automata. In this case the characteristic semigroup is left cancellative (Lemma 2). If a state-independent (quasi-) automaton is also A-finite, then its characteristic semigroup is a right group (see [9] or Lemma 3).

In 1966, Ch. A. Trauth ([8]) introduced the notion of the group-type automaton (state-independent automaton whose characteristic semigroup is a group) and characterized the quasi-perfect (strongly connected and group-type) automata. He proved that if $\mathbf{A}_i$ ($i \in I$) is a family of quasi-perfect (quasi-) automata and G_i ($i \in I$) is the family of corresponding characteristic semigroups, then a quasi-perfect (quasi-) automaton $\mathbf{A}$ is decomposable into an A-direct product of automata $\mathbf{A}_i$ if and only if the characteristic semigroup of $\mathbf{A}$ is a direct product of the groups G_i . In 1975, I. Babcsányi ([2]) dealt with the decomposition of group-type generated automata. He proved that every generated group-type quasi-automaton is a direct

*This work was lectured on Colloquium on Semigroups, Szeged, 15-19 August, 1994.,Research was supported by Hungarian National Foundation for Scientific Research grant No 7608.

†Department of Mathematics, Transport Engineering Faculty, Technical University of Budapest, 1111 Budapest, Műegyetem rkp. 9., HUNGARY

sum of pairwise isomorphic quasi-perfect quasi-automata. In 1976, Y. Masunaga, S. Noguchi and J. Oizumi ([7]) proved that every strongly connected state-independent A-finite (quasi-) automaton is isomorphic to a A-direct product of a quasi-perfect (quasi-) automaton and a strongly connected reset (quasi-) automaton.

In this paper we extend the investigations to the (not necessarily A-finite) state-independent automata whose characteristic semigroups are right groups.

For notations and notions not defined here, we refer to [4] and [6].

Let $\mathbf{A} = (A, X, \delta)$ be an arbitrary automaton. We suppose that the transition function δ is extended to $A \times X^+$ (X^+ denotes the free semigroup over X) as usually, that is, $\delta(a, px) = \delta(\delta(a,p), x)$ ($p \in X^+$, $x \in X$). For brevity, let $\delta(a,p)$ be denoted by ap. For an arbitrary automaton $\mathbf{A} = (A, X, \delta)$, we consider the following quasi-automata $\mathbf{A}^* = (A, S(A), \delta^*)$ and $\mathbf{A}^*_S = (AS(A), S(A), \delta^*)$, where $S(A)$ is the characteristic semigroup of $\mathbf{A}$, δ^* is defined by $\delta^*(a, \overline{p}) = \delta(a,p)$ $(a \in A,\ p \in X^+)$ and $AS(A) = \{\delta^*(a,s);\ a \in A, s \in S(A)\}$. $\mathbf{A}^*_S$ will be called the *quasi-automaton corresponding to the automaton* $\mathbf{A}$.

Definition 1. An automaton or a quasi-automaton $\mathbf{A}$ is called a *(right) group-type automaton* if it is state-independent and $S(A)$ is a (right) group.

It is clear that an automaton $\mathbf{A}$ is state-independent if and only if $\mathbf{A}^*$ is state-independent. As $S(A) \cong S(A^*)$, it follows that $\mathbf{A}$ is a (right) group-type automaton if and only if $\mathbf{A}^*$ is (right) group-type.

Definition 2. Let $\{S_e :\ e \in E\}$ be an E right zero semigroup decomposition of a semigroup S, that is, E is a right zero semigroup and S is a disjoint union of its subsemigroups S_e, $e \in E$ such that $S_eS_f \subset S_{ef} = S_f$, for every $e, f \in E$. We say that a quasi- automaton $\mathbf{A} = (A, S, \delta)$ is a *right zero decomposition* of quasi-automata $\mathbf{A}_e = (A_e, S_e, \delta_e)$ $(e \in E)$ with $A_e \cap A_f = \emptyset$ for all $e \neq f \in E$, if $A = \cup_{e \in E} A_e$ and $AS_e = \{\delta(a,s) :\ a \in A, s \in S_e\} \subseteq A_e$.

Lemma 1. *A state-independent automaton* $\mathbf{A}$ *is right group-type if and only if the quasi-automaton* $\mathbf{A}^*_S$ *corresponding to* $\mathbf{A}$ *is right group-type.*

Proof. Let $\mathbf{A}$ be a state-independent automaton. Then $\mathbf{A}^*$ and so $\mathbf{A}^*_S$ is state-independent. Moreover, $S(A) \cong S(A^*) = S(A^*_S)$. If $\mathbf{A}$ is right group-type, then $\mathbf{A}^*_S$ is right group-type, too.

Conversely, let $\mathbf{A}^*_S$ be right group-type. As $\mathbf{A}^*$ is state-independent and $S(A^*) = S(A^*_S)$, we get that $S(A^*)$ is a right group. As $S(A) \cong S(A^*)$, the automaton $\mathbf{A}$ is right group-type. □

Theorem 1. *A state-independent automaton* $\mathbf{A}$ *is right group-type if and only if the quasi-automaton* $\mathbf{A}^*_S$ *corresponding to* $\mathbf{A}$ *is a right zero decomposition of pairwise isomorphic group-type quasi-automata.*

Proof. Let the state-independent automaton $\mathbf{A}$ be right group-type. Then, by Lemma 1, the quasi-automaton $\mathbf{A}^*_S$ corresponding to $\mathbf{A}$ is right group-type. Since $S(A^*_S)$ is a right group, it is a right zero semigroup E of its subgroups G_e, where $G_e = Ge$ for some subgroup G of $S(A^*_S)$. Let $A_e = AG_e$, $e \in E$. It is evident that $\mathbf{A}_e = (A_e, G_e, \delta_e)$ are group-type qusi-automata. We show that $A_e \cap A_f = \emptyset$ if $e \neq f$. Let us suppose that $age = bhf \in A_e \cap A_f$ for some $a, b \in A$, $g, h \in G$ and $e, f \in E$. Then $agf = bhf$ from which it follows that $age = agf$. As $\mathbf{A}$ is state-independent we have $e = f$. Hence $A_e = A_f$. It is evident that $AG_e \subseteq A_e$ and $A_S^* = \cup_{e \in E} A_e$. Consequently, $\mathbf{A}_S^*$ is a right zero decomposition of the group-type quasi-automata $\mathbf{A}_e$, $e \in E$. To complete the proof we show that the quasi-automata $\mathbf{A}_e, e \in E$ are isomorphic with each other. Let $\alpha_{e,f} : A_e \to A_f$ and $\beta_{e,f} : G_e \to G_f$ defined by

$$\alpha_{e,f}(age) = agf, \quad \beta_{e,f}(ge) = gf, \quad a \in A, \ g \in G.$$

It is easy to check that $(\alpha_{e,f}, \beta_{e,f})$ is an isomorphism of $\mathbf{A}_e$ onto $\mathbf{A}_f$.

Conversely, assume that $\mathbf{A}_S^*$ is a right zero decomposition of pairwise isomorphic group-type quasi-automata $\mathbf{A}_e = (A_e, G_e, \delta_e)$, $e \in E$. Then it is easy to see that $S(A_S^*)$ is a right group, and so, $\mathbf{A}_S^*$ is right group-type. Therefore, by Lemma 1, we obtain that $\mathbf{A}$ is right group-type. □

The following example shows that if an automaton $\mathbf{A}$ is right group-type then it is not necessarily a right zero decomposition of pairwise isomorphic group-type automata.

Example 1. Let the state-independent automaton $\mathbf{A} = (\mathbf{A}, \mathbf{X}, \delta)$ be the direct sum of the automata $\mathbf{A_1} = (\mathbf{A_1}, \mathbf{X}, \delta_1)$ and $\mathbf{A_2} = (\mathbf{A_2}, \mathbf{X}, \delta_2)$ $((A_1 = \{1,2,3,4,5\}, (A_2 = \{6,7,8,9,10,11\}, X = \{x, y, z\})$ which are defined by the following transition tables:

$\mathbf{A_1}$	1	2	3	4	5
x	2	3	2	2	3
y	3	2	3	3	2
z	5	4	5	5	4

$\mathbf{A_2}$	6	7	8	9	10	11
x	7	8	7	7	8	7
y	8	7	8	8	7	8
z	10	9	10	10	9	10

The Cayley-table of the characteristic semigroup $S(A)$:

	$\overline{x}$	$\overline{y}$	$\overline{z}$	$\overline{z^2}$
$\overline{x}$	$\overline{y}$	$\overline{x}$	$\overline{z^2}$	$\overline{z}$
$\overline{y}$	$\overline{x}$	$\overline{y}$	$\overline{z}$	$\overline{z^2}$
$\overline{z}$	$\overline{y}$	$\overline{x}$	$\overline{z}^2$	$\overline{z}$
$\overline{z^2}$	$\overline{x}$	$\overline{y}$	$\overline{z}$	$\overline{z^2}$

The quasi-automaton $\mathbf{A}_S^*$ is a direct sum of the quasi-automata $\mathbf{A}_{1S}^*$ and $\mathbf{A}_{2S}^*$ given by the following transition tables:

$\mathbf{A}_{1S}^*$	2	3	4	5
$\overline{x}$	3	2	2	3
$\overline{y}$	2	3	3	2
$\overline{z}$	4	5	5	4
$\overline{z^2}$	5	4	4	5

$\mathbf{A}_{2S}^*$	7	8	9	10
$\overline{x}$	8	7	7	8
$\overline{y}$	7	8	8	7
$\overline{z}$	9	10	10	9
$\overline{z^2}$	10	9	9	10

It is easy to check that $\mathbf{A}_S^*$ is right group-type and is a right zero decomposition of the group-type quasi-automata $\mathbf{B}_1$ and $\mathbf{B}_2$ given by the following transtion tables. (We note that $\{S(B_1), S(B_2)\}$ is a right zero semigroup decomposition of $S(A)$.)

$\mathbf{B}_1$	2	3	7	8
$\overline{x}$	3	2	8	7
$\overline{y}$	2	3	7	8

$\mathbf{B}_2$	4	5	9	10
$\overline{z}$	5	4	10	9
$\overline{z^2}$	4	5	9	10

Lemma 2. ([3]) *The characteristic semigroup of a state-independent quasi-automaton is left cancellative.*

Lemma 3. *An A-finite automaton is state-independent if and only if it is right group-type.*

Proof. Let $\mathbf{A}$ be an A-finite state-independent automaton. Then, by Lemma 2, $S(A)$ is a (finite) left cancellative semigroup. It is easy to show that $S(A)$ is also right simple. Hence $S(A)$ is a right group, that is $\mathbf{A}$ is a right group-type automaton. The converse statement follows from the definition. □

The following example shows that the assertion of Lemma 3 is not true in infinite case.

Example 2. Let $\mathbf{A} = (A, X, \delta)$ be an automaton where A is the set of all positive integers, $X = \{x\}$ and δ is defined by $\delta(n, x) = n + 1$ $(n \in A)$. It is easy to see that $\mathbf{A}$ is state-independent whose characteristic semigroup is an infinite cyclic semigroup.

Lemma 4. *Every group-type quasi-automaton $\mathbf{A}_S^*$ corresponding to a state-independent automaton $\mathbf{A}$ is a direct sum of pairwise isomorphic quasi-perfect quasi-automata.*

Proof. See Lemma 2 and Lemma 4 of [2]. □

The following theorem is a generalization of Lemma 4 for right group-type (quasi-) automata.

Theorem 2. *A state-independent automaton $\mathbf{A}$ is right group-type if and only if the quasi-automaton $\mathbf{A}_S^*$ corresponding to $\mathbf{A}$ is a direct sum of pairwise isomorphic strongly connected right group-type quasi-automata.*

Proof. Let the state-independent automaton $\mathbf{A}$ be right group-type. Then, by Lemma 1, the quasi-automaton $\mathbf{A}_S^*$ corresponding to $\mathbf{A}$ be right group-type. For an arbitrary $a \in AS(A)$, we consider the following A-subautomaton $\mathbf{A}(a) = (A(a), S(A), \delta_a)$ of $\mathbf{A}_S^*$, where $A(a) = \{as : \ s \in S(A)\}$. As $S(A)$ is a right group, therefore $\mathbf{A}(a)$ is strongly connected. As every A-subautomaton of a state-independent (quasi-) automaton $\mathbf{A}$ is also state-independent such that its characteristic semigroup is $S(A)$, we get that $\mathbf{A}(a)$ is a right group-type automaton. It is easy to see that $A(a) \cap A(b) \neq \emptyset$ implies $A(a) = A(b)$ for every $a, b \in AS(A)$. Moreover $as \to bs$ $(a, b \in AS(A),\ s \in S(A))$ is an isomorphism of $\mathbf{A}(a)$ onto $\mathbf{A}(b)$.

Thus $\mathbf{A}_S^*$ is a direct sum of the pairwise isomorphic different A-subautomata $\mathbf{A}(a)$. The converse statement of the theorem is evident. □

We note that the quasi-automaton $\mathbf{A}_S^*$ considered in Example 1 is a direct sum of isomorphic strongly connected right group-type quasi-automata $\mathbf{A}_{1S}$ and $\mathbf{A}_{2S}$. It shows that the components of the direct sum are different from the components of the right zero decomposition.

Lemma 5. *If a quasi-automaton* $\mathbf{A} = (A, S, \delta)$ *is quasi-perfect, then* $|A| = |S(A)|$ (see Lemma 6 and Theorem 3 of [1]).

Corollary 1. *If* $\mathbf{A}$ *is an A-finite state-independent automaton, then* $|S(A)|$ *is a divisor of* $|AS(A)|$.

Proof. Let $\mathbf{A}$ be an A-finite state-independent automaton. Then, by Lemma 3, $\mathbf{A}$ is right group-type. By Lemma 1 and Theorem 1, $\mathbf{A}_S^*$ is a right zero decomposition of pairwise isomorphic group-type quasi-automata $\mathbf{A}_e = (A_e, G_e, \delta_e)$, $e \in E$. Then $|AS(A)| = |A_e||E|$ for arbitrary $e \in E$. By Lemma 4 and Lemma 5, $|A_e| = n|G_e|$ for some positive integer n. Hence $|AS(A)| = n|G_e||E| = n|S(A)|$. □

Corollary 2. *The quasi-automaton* $\mathbf{A}_S^*$ *corresponding to an A-finite state-independent automaton* $\mathbf{A}$ *is a right zero decomposition of pairwise isomorphic quasi-perfect quasi-automata if and only if* $|AS(A)| = |S(A)|$.

Proof. Let the quasi-automaton $\mathbf{A}_S^*$ corresponding to an A-finite state-independent automaton $\mathbf{A}$ be a right zero decomposition of pairwise isomorphic quasi-perfect quasi-automata $\mathbf{A}_e = (A_e, G_e, \delta_e)$, $e \in E$. By Lemma 5, $|A_e| = |G_e|$. Hence $|AS(A)| = |S(A)|$.

Conversely, let $\mathbf{A}$ be an A-finite state-independent automaton such that $|AS(A)| = |S(A)|$. By Lemma 3, $\mathbf{A}$ is right group-type. Then, by Lemma 1 and Theorem 1, $\mathbf{A}_S^*$ is a right zero decomposition of pairwise isomorphic group-type quasi-automata $\mathbf{A}_e = (A_e, G_e, \delta_e)$, $e \in E$. (Here $G_e = Ge$, for some subgroup G of $S(A)$, and $A_e = AG_e$.) It is sufficient to show that $\mathbf{A}_e$ are strongly connected. It is evident that $|S(A)| = |G||E|$ and $|AS(A)| = |A_e||E|$, for every $e \in E$. Then $|A_e| = |G|$, for every $e \in E$. As $\mathbf{A}$ is state-independent, we have $|aG_e| = |G| = |A_e|$, for every $e \in E$ and $a \in A_e$. From this it follows that $\mathbf{A}_e$ is strongly connected, for every $e \in E$. □

We note that the quasi-automata $\mathbf{A}_{1S}^*$ and $\mathbf{A}_{2S}^*$ in Example 1 satisfy the conditions of Corollary 2. For example, the quasi-perfect components of the right zero decomposition of $\mathbf{A}_{1S}^*$ are:

$\mathbf{A}_3$	2 3
$\overline{x}$	3 2
$\overline{y}$	2 3

$\mathbf{A}_4$	4 5
$\overline{z}$	5 4
$\overline{z^2}$	4 5

It is easy to check that these components are isomorphic.

References

[1] Babcsányi, I., *A félperfekt kváziautomatákról (On quasi-perfect quasi-automata)*, Mat. Lapok, 21, 1970, 95–102 (in Hungarian with English summary)

[2] Babcsányi, I., *Endomorphisms of group-type quasi-automata*, Acta Cybernetica, 2, 1975, 313–322

[3] Babcsányi, I., *Characteristically free quasi-automata*, Acta Cybernetica, 3, 1977, 145–161

[4] Clifford, A.H. and G.B. Preston, *The Algebraic Theory of Semigroups*, Amer. Math. Soc., Providence, I(1961), II(1967)

[5] Fleck, A.C., *On the automorphism group of an automaton*, J. Assoc. Comp. Machinery, 12, 1965, 566-569

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

[7] Masunaga, Y., S. Noguchi and J. Oizumi, *A characterization of automata and a direct product decompostion*, Journal of Computer and System Sciences, 13, 1976, 74–89

[8] Trauth, Ch. A., *Group-type automata*, J. Assoc. Comp. Mach., 13, 1966, 170–175

[9] Watanabe, T. and S. Noguchi, *The amalgamation of automata*, Journal of Computer and System Sciences, 15, 1977, 1–16

Received January, 1995

Acta Cybernetica 12 (1995) 137-143.

Some Properties of H–functions

Ivan Mirchev* Borislav Yurukov*

1 Introduction

Some basic results in the theory of separable and c–separable sets were obtained in [1]-[7]. In this paper some problems concerning with separable and c–separable sets for k–valued functions are considered.

We investingate the properties of k–valued functions when some of their variables are replaced with constants. The investigations of properties of H–functions are connected with separability and c–separability of functions.

2 Definitions and Notations

Definition 1 [1] A function $f(x_1,...,x_n)$ on $A(|A| \geq 2)$ *depends essentially on the variable* x_i, $1 \leq i \leq n$ if there exist $n-1$ constants $c_1,...,c_{i-1},c_{i+1},...,c_n$ such that the unary function $f(c_1,...,c_{i-1},x,c_{i+1},...,c_n)$ takes on at least two different values.

Ess(f) denotes the set of all variables which f depends essentially on.

$\mathbf{F_n}$ denotes the set of all functions which depend essentially exactly on n variables.

Definition 2 [1] A function f and the functions obtained from f by replacing some of its variables with constants are called subfunctions of f ($g \longrightarrow\!\!\!< f$ denotes that g is a subfunction of f).

Definition 3 [4] The variable x_i, $1 \leq i \leq n$, $n \geq 1$ is a *H–variable* for a function $f \in F_n$ if for any two tuples of constants differing only in the i^{-th} component, the function has different values.

Definition 4 [4] The function f is a H–function if all its essential variables are H–variables.

$\mathbf{H^k_{f_n}}$ denotes the set of all k–valued H–functions from F_n. $\mathbf{H^k_f}$ denotes the set of all k–valued H–functions.

*Faculty of Mathematics and Natural Sciences "Neofit Rilski" South-West University 66 "Al. Velichkov" Str., 2700 Blagoevgrad, Bulgaria Fax. 00359/73/29325, Tel. 00359/73/26174, 20767

3 Basic Results

The following assertion is obvious.

Statement 1 *A function $f \in F_n$, $n \geq 2$ is a H–function if and only if all of its subfunctions from F_k, $1 \leq k < n$ are H–functions too.*

Theorem 1 *Let $p \geq 3$ be a prime number and let $f \in F_n$, $n \geq 2$, be a non-linear p–valued function. If there exists f_1, $f_1 \longrightarrow\!\!\!< f$, $|Ess(f_1)| = 1$ which as polynomial mod p is of degree $p-1$ then $f \notin H^k_{f_n}$.*

Proof. By Statement 1 it is sufficient to prove that every polynomial

$$f_1(x) = a_0 + a_1 x + \dots + a_{p-1}x^{p-1} \pmod p,\ a_{p-1} \neq 0$$

cannot take on all values from the set $\{0, 1, \dots, p-1\}$. Consider the polynomial

$$g(x) = a_1 x + a_2 x^2 + \dots + a_{p-1}x^{p-1} \pmod p,\ a_{p-1} \neq 0.$$

Let us assume that

$$g(i) = b_i,\ i = 1, 2, \dots, p-1,\ b_i \neq b_j \text{ when } i \neq j \text{ and } b_i \neq 0, \text{ if } i \neq 0.$$

The determinant of the system

$$a_1 i + a_2 i^2 + \dots + a_{p-1} i^{p-1} = b_i,\ i = 1, 2, \dots, p-1$$

is

$$\Delta = \begin{vmatrix} 1 & 1^2 & \dots & 1^{p-1} \\ 2 & 2^2 & \dots & 2^{p-1} \\ 3 & 3^2 & \dots & 3^{p-1} \\ . & . & . & . \\ (p-1) & (p-1)^2 & \dots & (p-1)^{p-1} \end{vmatrix} = 1.2 \dots (p-1).W(1, 2, \dots, p-1).$$

Using the facts that

$$W(c_1, \dots, c_k) = \prod_{k \geq i \geq j \geq 1} (c_i - c_j)$$

and

$$(p-1)! + 1 \equiv 0 \pmod p, \quad \text{we have} \quad \Delta \neq 0.$$

Consequently the system has only one solution. As we know $a_{p-1} = \frac{\Delta_{p-1}}{\Delta}$, where

$$\Delta_{p-1} = \begin{vmatrix} 1 & 1^2 & \dots & 1^{p-2} & b_1 \\ 2 & 2^2 & \dots & 2^{p-2} & b_2 \\ 3 & 3^2 & \dots & 3^{p-2} & b_3 \\ . & . & . & . & . \\ (p-1) & (p-1)^2 & \dots & (p-1)^{p-1} & b_{p-1} \end{vmatrix}.$$

But

$$\Delta_{p-1} = \begin{vmatrix} 1 & 1^2 & \dots & 1^{p-2} & b_1 \\ 2 & 2^2 & \dots & 2^{p-2} & b_2 \\ 3 & 3^2 & \dots & 3^{p-2} & b_3 \\ . & . & . & . & . \\ S_1 & S_2 & \dots & S_{p-2} & S \end{vmatrix}, \text{ where}$$

$$S_k = 1^k + 2^k + \dots + (p-1)^k, \; k = 1, 2, \dots, (p-2),$$

$$S = b_1 + b_2 + \dots + b_{p-1} = 1 + 2 + 3 + \dots + (p-1) = S_1.$$

The numbers $1, 2, \dots, p-1$ are solutions of the equation $x^{p-1} - 1 = 0 \pmod p$. Consequently for the elementary symmetric polynomials $\tau_1, \tau_2, \dots, \tau_{p-2}$ of $1, 2, \dots, p-1$ we have

$$\tau_1 = \tau_2 = \dots = \tau_{p-2} = 0.$$

On the other hand from Newton's formulas

$$S_k - \tau_1.S_{k-1} + \tau_2.S_{k-2} - \dots + (-1)^{k-1}\tau_{k-1}.S_1 + (-1)^k.k\tau_k = 0,$$

when $k \leq p-1$.

If $k < p-1$, then $S_k = 0$. Consequently $\Delta_{p-1} = 0$ implies $a_{p-1} = 0$. This contradicts the condition $a_{p-1} \neq 0$.

Therefore the values of the polynomials $g(x)$ and $f_1(x)$ cannot form a whole system modulo p. This completes the proof. □

Remarks:

1. If $p = 2$, then according to Lemma 4.2 [3], Theorem 4.1 [3] and Lemma 4.10 [3] it follows that $f \in H^2_{f_n}$ if and only if f is a linear function.

2. When $p = 3$ this theorem was proved by K. Chimev in [4] and now was improved (by Mirchev and Drenski) for $p \geq 3$, where p – a prime number.

3. It is obvious that if $f \in L_p$ then $f \in H^p_f$ (L_p denotes the set of all linear p–valued functions). The converse statement is not valid and this fact is evident from the following example.

Example 1 Let $f(x_1, x_2) = x_1^3 + x_2^3 \pmod 5$. For the function f, $f \in H^5_{f_2}$ but $f \notin L_5$ (Here $x_i^3 = x_i.x_i.x_i,\; i = 1, 2$).

Now we will consider some results which give us good possibilities to construct catalogues of H–functions modulo 3.

Definition 5 We will say that $f(x_1, \dots, x_n)$ and $g(x_1, \dots, x_n)$ are *distinguishable everywhere* if for each tuple of constants $c_1, \dots, c_n$ the relation

$$f(c_1, \dots, c_n) \neq g(c_1, \dots, c_n) \text{ holds.}$$

We denote by $f <> g$ that f and g are distinguishable everywhere.

Let $j_i(x) = \begin{cases} 1, & x = i, \\ 0, & x \neq i. \end{cases}$

If $f(x_1, ..., x_n)$, $n \geq 2$, is a k–valued function then it is obvious that $\forall p (1 \leq p \leq n)$

$$f(x_1, ..., x_n) = \sum_{i=0}^{k-1} j_i(x_p).f(x_1, ..., x_{p-1}, i, x_{p+1}, ..., x_n).$$

If
$$\begin{aligned} f_1(x_1, ..., x_{n-1}) &= f(x_1, ..., x_{n-1}, 0), \\ &\vdots \\ f_k(x_1, ..., x_{n-1}) &= f(x_1, ..., x_{n-1}, k-1), \end{aligned}$$
then
$$f(x_1, ..., x_n) = \sum_{i=0}^{n} j_i(x_n) f_{i+1}(x_1, ..., x_{n-1}).$$

Theorem 2 (Theorem 1 [6]) *$f \in H^k_{f_n}$, $n \geq 2$, if and only if $f_i \in H^k_{f_{n-1}}$ and $f_i <> f_j$ for $i, j = 1, ..., k$, $i \neq j$.*

According to this result each function $f \in H^3_{f_n}$ can be derived from f_1, f_2, f_3, where for each $1 \leq i \leq 3$, $1 \leq j \leq 3$ and $i \neq j$ the relations

$$f_i \in H^3_{f_{n-1}} \text{ and } f_i <> f_j \text{ hold.}$$

We denote by $f = (f_1, f_2, f_3)$ the fact that $f \in H^3_{f_n}$ is derived from $f_1, f_2, f_3 \in H^3_{f_{n-1}}$.

Lemma 1 *Let $f = (f_1, f_2, f_3)$, $g = (g_1, g_2, g_3)$ and $f, g \in H^3_{f_n}$. Then $f <> g$ if and only if $f_1 <> g_1$, $f_2 <> g_2$ and $f_3 <> g_3$.*

Proof.

" $\Rightarrow$ " Let $f <> g$. Then $f(x_1, ..., x_{n-1}, 0) <> g(x_1, ..., x_{n-1}, 0)$, i.e. $f_1 <> g_1$. Analogously $f_2 <> g_2$ and $f_3 <> g_3$.

" $\Leftarrow$ " Let $f_i <> g_i$, $i = 1, 2, 3$. Let us suppose that there exist $c_1, ..., c_n$ so that $f(c_1, ..., c_n) = g(c_1, ..., c_n)$.

If $c_n = 0$, then we obtain $f_1(c_1, ..., c_{n-1}) = g_1(c_1, ..., c_{n-1})$ which contradicts the condition $f_1 <> g_1$.

If $c_n = 1$ or $c_n = 2$ we obtain a contradiction with $f_2 <> g_2$ or $f_3 <> g_3$. □

Theorem 3 *If $f \in H^3_{f_n}$, $n \geq 2$ then there exist g and h, $g <> h$ and g, $h \in H^3_{f_n}$, such that $f <> g$ and $f <> h$.*

Proof.

$$\text{Let } f = (f_1, f_2, f_3); \quad g = (f_2, f_3, f_1); \quad \text{and } h = (f_3, f_1, f_2).$$

Since f_1, f_2, f_3 are pairwise distinguishable everywhere then according to Lemma 1, f, g and h are pairwise distinguishable everywhere too. By Theorem 2 we have

$$g \in H^3_{f_n} \text{ and } h \in H^3_{f_n}. \qquad \square$$

Theorem 4 *If $f \in H^3_{f_n}$ then there exist only two functions $g, h \in H^3_{f_n}$, such that f, g and h are pairwise distinguishable everywhere.*

Proof. We will prove the theorem by induction on the number of the variables.

The case $n = 1$ is trivial.

Let us assume that for functions from F_{n-1} the statement is true.

Let now $f \in H^3_{f_n}$. By Theorem 3, it is sufficient to prove that there exist only two functions g and h.

Let

$$f = (f_1, f_2, f_3), \text{ where } f_i <> f_j \text{ when } i \neq j;$$

$$g = (g_1, g_2, g_3), \text{ where } g_i <> g_j \text{ when } i \neq j;$$

$$h = (h_1, h_2, h_3), \text{ where } h_i <> h_j \text{ when } i \neq j;$$

$g, h \in H^3_{f_n}$, $g <> f$, $h <> f$, $g <> h$ and $f_i, g_i, h_i \in H^3_{f_{n-1}}$, $1 \leq i$, $j \leq 3$.

Since f, g and h are pairwise distinguishable everywhere then according to Lemma 1, f_1, g_1 and h_1 are distinguishable everywhere too.

By the induction hypothesis on f_1 there exist only two functions which are distinguishable everywhere from f_1. Therefore $\{g_1, h_1\} = \{f_2, f_3\}$.

Similarly we get:

$$\{g_2, h_2\} = \{f_1, f_3\}, \{g_3, h_3\} = \{f_1, f_2\}. \tag{1}$$

Withoutloss of generality we can assume that

$$g_1 = f_2 \text{ and } h_1 = f_3. \tag{2}$$

If we suppose $h_2 = f_3$, then from $h_1 = f_3$ we obtain $h_1 = h_2$, which contradicts the condition $h_1 <> h_2$. Therefore from (1) we obtain

$$g_2 = f_3 \text{ and } h_2 = f_1. \tag{3}$$

If we assume $g_3 = f_2$, then from $g_1 = f_2$ we obtain $g_1 = g_3$, which contradicts the condition $g_1 <> g_3$. Therefore from (1)

$$g_3 = f_1 \text{ and } h_3 = f_2. \tag{4}$$

Consequently g and h are exactly determined by f. $\square$

Theorem 5 *If $f, g, h \in H^3_{f_n}$, $g \neq h$, $f <> g$ and $f <> h$ then $g <> h$.*

Proof. We will prove the theorem by induction on the number of the variables.

Let $n = 1$. Then:

$$f(0) = a_1,\ f(1) = a_2,\ f(2) = a_3,\ g(0) = a'_1,\ g(1) = a'_2,\ g(2) = a'_3,$$

$$h(0) = a_1'', \ h(1) = a_2'', \ h(2) = a_3'', \text{ i. e.}$$

$$f = (a_1 \ a_2 \ a_3), \text{ where } a_i \neq a_j \text{ when } i \neq j;$$

$$g = (a_1' \ a_2' \ a_3'), \text{ where } a_i' \neq a_j' \text{ when } i \neq j;$$

$$h = (a_1'' \ a_2'' \ a_3''), \text{ where } a_i'' \neq a_j'' \text{ when } i \neq j.$$

Let us assume that $g <> h$ doesn't hold. Without loss of generality we may assume that $a_1' = a_1''$. Then $\{a_2', a_3'\} = \{a_2'', a_3''\}$.

If we suppose that $a_2' = a_2''$ then we get $a_3' = a_3''$. Therefore $g = h$ which is a contradiction.

Let us now suppose that $a_2' = a_3''$ and $a_3' = a_2''$, i.e., that

$$g = (a_1' \ a_2' \ a_3') \text{ and } h = (a_1' \ a_3' \ a_2').$$

But $a_2 \notin \{a_2', a_3'\}$ and $a_3 \notin \{a_3', a_2'\}$ therefore $a_2 = a_3$. This contradicts the condition $f \in H^3_{f_1}$.

So, if $n = 1$ the statement is true.

Let us assume that the statement is true for all functions from F_{n-1}. We will prove the statement for the functions from F_n, $n \geq 2$.

Let

$$f = (f_1, f_2, f_3), \text{ where } f_i <> f_j \text{ when } i \neq j;$$

$$g = (g_1, g_2, g_3), \text{ where } g_i <> g_j \text{ when } i \neq j;$$

$$h = (h_1, h_2, h_3), \text{ where } h_i <> h_j \text{ when } i \neq j$$

and $f_i, g_i, h_i \in H^3_{f_{n-1}}$ $(1 \leq i, j \leq 3)$. As we know $f <> g$, $f <> h$ and $g \neq h$. Consequently $g_1 \neq h_1$ or $g_2 \neq h_2$ or $g_3 \neq h_3$.

Whitout loss of generality we can assume that $g_1 \neq h_1$.

From the conditions of the Theorem we obtain $f_1 <> g_1$ and $f_1 <> h_1$. But $f_1, g_1, h_1 \in H^3_{f_{n-1}}$. From this fact and our inductive supposition it follows that

$$g_1 <> h_1. \tag{5}$$

Since f_1, f_2, f_3 and f_1, g_1, h_1 are pairwise distinguishable everywhere it follows from Theorem 4 that

$$\{f_2, f_3\} = \{g_1, h_1\}.$$

Let us assume now that

$$g_1 = f_2 \text{ and } h_1 = f_3.$$

Since g_1, g_2, g_3 and f_1, f_2, f_3 are pairwise distinguishable everywhere and $g_1 = f_2$ it follows from Theorem 4 that

$$\{g_2, g_3\} = \{f_1, f_3\}.$$

Similarly as above, we have

$$\{h_2, h_3\} = \{f_1, f_2\}.$$

If we suppose that $g_2 = h_2$ or $g_2 = h_3$ then we obtain

$$g_2 \in \{f_1, f_3\} \cap \{f_1, f_2\} = \{f_1\}.$$

Therefore $g_2 = f_1$, $g_3 = f_3$, which contradicts $g_3 <> f_3$.

If we suppose that $g_3 = h_3 = f_1$ then we have $h_2 = f_2$, which contradicts $h_2 <> f_2$. Consequently $g_3 = h_2 = f_1$, $g_2 = f_3$, $h_3 = f_2$ which implies

$$g_2 <> h_2 \text{ and } g_3 <> h_3. \tag{6}$$

From Lemma 1, (5) and (6) it follows that $g <> h$. □

Finally we note, that some algorithms, computer programs and catalogues for H–functions are given in [3].

References

[1] Chimev, K.N., Separable Sets of Arguments of Functions, MTA SZTAKI, Tanulmanyok 180/1986. 1-173 p.

[2] Chimev, K.N., Funkcii i grafi [Functions and Graphs], South-West University Publishing House, Blagoevgrad, 1983.

[3] Mirchev, I. A. Otdelimi i dominirashti mnojestva ot promenlivi - disertacija [Separable and Dominating Sets of Variables - dissertation], Sofia University - Math, 1990.

[4] Chimev, K.N., Varhu edin nachin na zavisimost na njakoi funkcii ot P_k ot tehnite argumenti [On a Dependence of Some Functions from P_k on their Variables], Year-book of the Technical Universities, Math, v. IV, book 1, 1967, 5-12 p.

[5] Shtrakov, S. V., Dominating and Annuling Sets of Variables for the Functions, Blagoevgrad, 1987, 1-179 p.

[6] Gyudjenov, I.D., B.P. Yurukov, Varhu zavisimostta na funkciite ot tehnite promenlivi po shesti nachin [On the Sixth Way's Dependence of Some Functions on their Variables], Year-book of the South-West University, Math, book 1, 1987, 5-12 p.

[7] Yablonskii, S. V., O.B. Lupanov, Diskretnaya matematika i matematicheskie voprosi kibernetiki [Discrete mathematics and mathematical tasks of cybernetics], Moscow, 1974.

Received April, 1994

Acta Cybernetica 12 (1995) 145-152.

Quasioptimal Bound for the Length of Reset Words for Regular Automata

I.K. Rystsov *

1 Introduction

In 1964 J. Cerny stated the hypothesis that in a finite reset automaton with n states there is a reset (synchronizing) word whose length is at most $(n-1)^2$ and showed that this bound can be achieved [1]. In [2] this hypothesis was proved by direct enumeration of automata with small number of states. J. Pin used algebraic methods to prove this hypothesis for cyclic automata with prime number of states [3]. The general upper bound $(n^3-n)/6$ has been obtained in [4] for any reset n-state automaton.

The aim of this paper is to obtain the quasioptimal bound $2\cdot(n-1)^2$ for regular reset automata with n states and to extend the class of automata for which the optimal bound is valid.

2 Basic notions

A *finite deterministic automaton* A is a function $A : S \times X \to S$, where S is a *nonempty* finite set of states and X is a finite alphabet of input letters. This function can be considered as a function from X to the multiplicative monoid $Map(S)$ of unary mappings on S. So it can be naturally extended to a homomorphism from the free monoid X^* of words generated by X to the monoid $Map(S)$:

$$A: X^* \to Map(S).$$

This homomorphism associates with a word $w = x_1 \dots x_m$ the composition of mappings $A(w) = A(x_1)\cdot \dots \cdot A(x_m)$. Note that the empty word is mapped to the identical mapping. The submonoid $A(X^*)$ of $Map(S)$ is called the monoid of the automaton A.

Denote by $A(s,w)$ the value of the mapping $A(w)$ in the state $s \in S$. For a subset of states $T \subseteq S$ let us define $A(T,w) = \{A(s,w) \mid s \in T\}$. The *rank* of a word w with respect to A is equal to the number of states in the subset $A(S,w)$. A word is said to be *reset* for A if its rank with respect to A is equal to one . An

*Glushkov Institute of Cybernetics, Kiev, GSP, 252650, Ukraine

automaton is called *reset* if there is a reset word for it. The following proposition is evident.

Proposition 1 A finite automaton A is reset if and only if, for every pair of states s, t, there is a word w such that $A(s, w) = A(t, w)$.

For a word $w = x_1 \dots x_m$, $l(w) = m$ denotes its length. The set of all input words of length less than m is denoted by X_m. A finite nonempty set of input words will be called a *collection*. The length $l(W)$ of a collection W is the length of a longest word in it.

Let n be the number of states in A. A collection W is *transitive* for A if its length is less than n and for every pair of states s, t there is a word $w \in W$ such that $A(s, w) = t$. An automaton is said to be *transitive (strongly-connected)* if there is a transitive collection for it. In the sequel we shall consider only transitive automata because it is sufficient to prove the Cerny's hypothesis for this class of automata [5].

Definition 1 A transitive collection of words W is called regular for A if it contains the empty word and there is a natural number $k \geq 1$ such that for every pair of states s, t, there are exactly k words in W which take the state s into the state t. The constant k will be called the regularity degree.

An automaton is called *regular* if there is a regular collection of words for it. For example, an automaton is regular if there is an input letter which cyclically permutates all its states. More generally, an automaton A with n states is regular if the subset of mappings $A(X_n)$ contains a regular subgroup of permutations. Note that a regular group of permutations is a (noncommutative) scheme of relations [6].

3 Directed automata

The preimage of a subset $T \subseteq S$ under the inverse action of a word w is defined in the following way:

$$A^o(T, w) = \{s \mid A(s, w) \in T\}.$$

The next proposition is evident.

Proposition 2 A word w is reset for A if and only if there is a state s for which $A^o(s, w) = S$.

The number of states in a subset T is denoted by $\mid T \mid$. A word w is said to be *increasing for a subset* T if $\mid A^o(T, w) \mid > \mid T \mid$. A subset of states is *proper* if it is nonempty and is not equal to S.

Definition 2 A collection of words W is called increasing for A if for any proper subset of states in A there is an increasing word in W.

An automaton is called *directed* if there is an increasing collection of words for it.

Theorem 1 An automaton is directed if and only if it is reset and transitive.

Proof. Let A be a reset transitive automaton and w be a reset word for A. Then $A(S, w) = \{s_1\}$, for some state s_1. From transitivity of A it follows that for any state $s_i \in S$, there is a word w_i such that $A(s_1, w_i) = s_i$, $1 \leq i \leq n$. Thus we have $A^o(s_i, ww_i) = S$, for all $1 \leq i \leq n$, hence the collection of words $\{ww_i \mid 1 \leq i \leq n\}$ will be increasing for A.

Conversely, let A be a directed automaton and W be an increasing collection for it. Let us fix any state s_1 as an initial state. Then there is an increasing word $w_1 \in W$ for the subset $\{s_1\}$. Let $S_1 = A^o(s_1, w_1)$. If the subset S_1 is proper then there is an increasing word $w_2 \in W$ for S_1 and we take $S_2 = A^o(S_1, w_2)$. This step can be repeated several times until the set S will be obtained. By construction, we have the following series:

$$1 <\mid S_1 \mid<\mid S_2 \mid< \ldots <\mid S_m \mid= n. \tag{1}$$

As the result we obtain the word $w = w_m \ldots w_1$ such that $A^o(s_1, w) = S$. So, by proposition 2, the word w is reset for A. It is also easy to see that A is transitive, because an initial state can be choosed arbitrarily. Thus the theorem is proved. □

Let $res(A)$ be the minimal length of reset words for a directed automaton A and $inc(A)$ be the minimum over the lengthes of increasing collections of words for A. Theorem 1 implies the following relationship between these functions.

Theorem 2 For any directed automaton A with $n > 1$ states, the inequality $res(A) \leq inc(A) \cdot (n-2) + 1$ is valid.

Proof. Let A be a directed automaton and W be an increasing collection for it of minimal length $inc(A)$. According to theorem 1 A is reset, therefore there is an input letter $x_1 \in X$ for which the mapping $A(x_1)$ is not bijective. Then there is a state s_1 such that $\mid A^o(s_1, x_1) \mid> 1$. Let us fix s_1 as an initial state and repeat the procedure from theorem 1 with $w_1 = x_1$. From (1) it follows that the length of the resulting reset word is at most $l(W) \cdot (n-2) + 1$. This completes the proof. □

This theorem shows that inequality $inc(A) \leq n$ implies Cherny's hypothesis. Since it is difficult to obtain this bound by combinatorial methods, in the next section, we shall use more powerful methods of linear algebra.

4 Linear extensions of automata

Let R be the field of real numbers and R^n the n-dimensional vector space over R. Denote by $\langle u, v\rangle$ the scalar product of vectors u and v in this space. The standard basis E in this space consists of binary vectors $e_i, 1 \leq i \leq n$, where the i-th component of e_i is equal to one and the others are zeros.

For a collection of vectors V, denote by $Af(V)$ its affine span which consists of affine linear combinations of vectors in V with real coefficients [7]. If a set of vectors is equal to its affine span, then it is called an *affine subspace* of R^n. The *dimension of an affine subspace* is defined as the dimension of the parallel linear subspace [7].

The sum of basic vectors will be called the *unit vector* $e = (1, \dots 1)$. This vector defines the linear function from R^n to R in the usual way $|v| = \langle e, v \rangle$. The unit vector belongs to the following hyperplane:

$$P^n = \{v \mid \langle e, v \rangle = n\},$$

which is an $(n-1)$-dimensional affine subspace of R^n.

We say that a collection of vectors $V \subset P^n$ is *complete* if $Af(V) = P^n$. The *centre* c(V) of a collection $V = \{v_1, \dots, v_m\}$ is defined by the formula:

$$c(V) = \frac{1}{m} \sum_{i=1}^{m} v_i.$$

A collection of vectors V is *central* if $c(V) = e$.

Definition 3 A collection of vectors is called balanced if it is complete and central.

Let A be a finite deterministic automaton with a set of states $S = \{s_1, \dots, s_n\}$. Then there is a one-to-one correspondence f between S and the standard basis E of the space R^n which is defined as follows $f(s_i) = e_i, 1 \leq i \leq n$. Note that e_i is the characteristic vector of the subset $\{s_i\}$.

Now we define an isomorphic automaton L_A on the set E by the formula $L_A(e_i, x) = e_j$ iff $A(s_i, x) = s_j$. Then we can extend the transition function to the whole linear space as follows:

$$L_A(\sum_{i=1}^{m} r_i \cdot e_i, x) = \sum_{i=1}^{m} r_i \cdot L_A(e_i, x).$$

Thus we obtain the linear automaton L_A which is called the *linear extension* of the automaton A over the field R.

In general case when the basis is fixed, a linear automaton can be considered as a function from X into the algebra $Mat_n(R)$ of $n \times n$ matrices over R. In our case every matrix $L_A(x)$ is binary and row-monomial, because A is deterministic. The element (i, j) of the matrix $L_A(x)$ is equal to one if $A(s_i, x) = s_j$, otherwise it is zero. The product of matrices $L_A(w) = L_A(x_1) \cdot \ldots \cdot L_A(x_m)$ corresponds to the input word $w = x_1 \dots x_m$. The value of the transition function $L_A(v, w)$ is equal to the product of the row-vector v and the matrix $L_A(w)$.

Let us fix the unit vector $e = (1, \dots, 1)$ as the initial state of the automaton L_A. The collection of vectors $L_A(e, W) = \{L_A(e, w) \mid w \in W\}$ is associated with a collection of words W. It is easy to see that $L_A(e, W) \subset P^n$ for any collection W. A collection of words W is called *complete (central, balanced)* for L_A if the collection of vectors $L_A(e, W)$ is complete (central, balanced).

The *product (concatenation)* of two collections of words W, Y is defined in the usual way $WY = \{wy \mid w \in W, y \in Y\}$. The following proposition is linear analog of well-known Moore's theorem [9].

Theorem 3 For any directed automaton A with n states, the collection of words X_n is complete for its linear extension L_A.

Proof. Let w be a reset word of length $res(A)$ and W be a transitive collection of words for A of length $n-1$. Then we have $L_A(e, \{w\}W) = n \cdot E$. So the collection of words X_m, where $m = res(A) + n$, is complete because it contains the complete subcollection $\{w\}W$.

Now let us consider the increasing sequence of affine subspaces $Af(L_A(e, X_i)), 1 \leq i \leq m$. Dimensions of these subspaces are less than n, so there is a positive integer $i < n$ such that $Af(L_A(e, X_i)) = Af(L_A(e, X_{i+1}))$. Hence, we conclude that $Af(L_A(e, X_i)) = Af(L_A(e, X_j))$, for all $j > i$. Therefore, we have

$$Af(L_A(e, X_i)) = Af(L_A(e, X_n)) = Af(L_A(e, X_m)) = P^n$$

and our theorem is proved. □

Let $f(T)$ be the binary characteristic vector of a subset $T \subseteq S$ of length n. Note that the number of states in T is equal to the scalar product $\langle e, f(T) \rangle$.

Lemma 1 If a collection of words W is complete for L_A, then for any proper subset T of S there is a word $w \in W$ satisfying $\langle L_A(e, w), f(T) \rangle \neq \mid T \mid$.

Proof. Consider the following hyperplane:

$$P(T) = \{v \mid \langle v, f(T) \rangle = \mid T \mid\}.$$

The intersection $Q = P(T) \cap P^n$ is a proper affine subspace of P^n because $f(T) \notin P^n$. Hence, $L_A(e, W) \not\subseteq Q$ since the collection of vectors $L_A(e, W)$ is complete. Thus the lemma is proved. □

The inverse transition on a vector e_i and a letter x in the automaton L_A is defined as the product of e_i and the transposed matrix $L_A(x)^o$. Note that the matrix $L_A(x)^o$ is column-monomial, and so, there is an isomorphism between inverse transitions in automata A and L_A which can be described for a subset T and a word w by the following formula:

$$f(A^o(T, w)) = f(T) \cdot L_A(w)^o. \tag{2}$$

There is also the following well-known relationship between the scalar product and inverse action of a matrix which holds for any vectors u, v and word w [8]:

$$\langle u \cdot L_A(w), v \rangle = \langle u, v \cdot L_A(w)^o \rangle. \tag{3}$$

Now we can prove one of the main theorem.

Theorem 4 If a collection of words is balanced for the linear automaton L_A, then it is increasing for A.

Proof. Let $W = \{w_1, \ldots, w_m\}$ be a balanced collection of words for L_A and T be a proper subset of states in A. Denote the collection of vectors $L_A(e, W)$ by V. By our assuption, we have the following equalities:

$$\langle c(V), f(T)\rangle = \langle e, f(T)\rangle = \mid T \mid .$$

By the definition of $c(V)$, we have the following property:

$$\sum_{i=1}^{m} \langle L_A(e, w_i), f(T)\rangle = m \cdot \mid T \mid . \tag{4}$$

Then by Lemma 4, we conclude that there is a word w_j in W for which the following inequlity holds:

$$\langle L_A(e, w_j), f(T)\rangle > \mid T \mid . \tag{5}$$

Indeed, in the opposite case we should have $\langle L_A(e, w_i), f(T)\rangle \leq \mid T \mid$, for all $i, 1 \leq i \leq m$. Then Lemma 4 implies a contradiction because in this case the left-hand side of (4) should be less than the right-hand side.

Properties (2) and (3) implies the following equalities:

$$\langle L_A(e, w_j), f(T)\rangle = \langle e, f(T) \cdot L_A(w_j)^o\rangle = \mid A^o(T, w_j) \mid .$$

So the inequality $\mid A^o(T, w_j) \mid > \mid T \mid$ will be hold for the word w_j satisfying (5). Thus the word w_j is increasing for the subset T, which completes the proof. □

5 Regular automata

Let A denote a regular reset automaton of n states. Let us fix a regular collection of words $Y = \{y_1, \ldots, y_m\}$ for this automaton with regularity degree $k \geq 1$. By definition 1, the parameters k, m, n satisfy the equality $k \cdot n = m$. Recall that the collection Y contains the empty word and $l(Y) < n$.

Consider the linear extension L_A of A over the field R. The bistochastic matrix each element of which is equal to $1/n$ is denoted by J_n. It is easy to see that the following matrix equality holds:

$$\frac{1}{m} \sum_{j=1}^{m} L_A(y_j) = J_n. \tag{6}$$

From this we obtain the next proposition.

Lemma 2 For any collection W, the collection of words WY is central for L_A.

Proof. Let $W = \{w_1, \dots, w_l\}$. If we multiply the equality (6) from left by the vector $c(L_A(e, W))$, then we get the following equality:

$$\frac{1}{lm} \sum_{i=1}^{l} \sum_{j=1}^{m} L_A(e, w_i y_j) = e.$$

Therefore, the lemma is proved. □

Now we can prove the main result.

Theorem 5 There is a reset word for A whose length is at most $2 \cdot (n-1)^2$.

Proof. Consider the collection of words $W = X_n Y$. Since the collection Y contains the empty word, we have the inclusion $X_n \subset W$. Hence, from Theorem 3 we conclude that the collection W is complete for L_A. Lemma 6 implies that the collection W is central, and so, it is balanced for L_A. Then by Theorem 5, we get that the collection W is increasing for A. So we have the following inequalities:

$$inc(A) \leq l(W) \leq l(X_n) + l(Y) \leq 2 \cdot (n-1).$$

Now using Theorem 2, we obtain the following bounds:

$$res(A) \leq 2 \cdot (n-1) \cdot (n-2) + 1 \leq 2 \cdot (n-1)^2.$$

Thus the theorem is proved. □

At last we give a sufficient condition which implies the validity of Cerny's hypothesis.

Theorem 6 If the collection of words XY is complete for the linear extension of A, then $res(A) \leq (n-1)^2$.

Proof. Indeed, by Lemma 6, the collection XY is central, and so, it is balanced. Then by Theorem 5 we conclude that $inc(A) \leq n$. Thus the required statement follows from Theorem 2.

6 Conclusion

Note that theorem 8 gives the largest class of automata for which the optimal bound is known, because cyclic automata from papers [1] and [3] satisfy its condition. It is interesting to study the following hypothesis.

Hypothesis. *Any transitive automaton is regular.*

If this hypothesis is valid, then from Theorem 7 it follows that the quasioptimal bound $2 \cdot (n-1)^2$ holds for any reset n-state automaton.

References

[1] Cerny, J., Poznamka k homogennym experimentom s konecnymy automatami, *Math. Fyz. Cas. SAV*, v.14 (1964), 208-215.

[2] Cerny, J., Pirica, A., Rosenauerova B. On directable automata, *Kybernetika*, v.7 (1971), 289-298.

[3] Pin, J.E., Sur un cas particulier de la conjecture de Cerny, Lect. Notes Comp. Sci., v.62 (1978), 345-352.

[4] Kljachko, A., Rystsov, I., Spivak M. Extremal combinatorial problem concerning the length of the reset word in a finite automaton, *Cybernetics*, v.23 (1987), 165-170. (Tranlated from Russian)

[5] Rystsov, I.K., Rank of a finite automaton, *Cybernetics and System Analysis*, v.28 (1992), 323-328. (Translated from Russian)

[6] Bannai, E., Ito, T., *Algebraic combinatorics (Association schemes)*, The Benjamin/Cummings Publishing Company, 1984.

[7] Brondsted, A., *An introduction to convex polytopes*, Springer-Verlag, 1983.

[8] Halmos, P.R., *Finite-dimentional vector space*, Van-Nostrand, 1963.

[9] Moore, E., Gedanken experiments with sequential machines, in: Automata Studies, Princeton University Press, 1956.

Received January, 1995

Acta Cybernetica 12 (1995) 153-166.

Fuzzy Extension of Datalog*

Ágnes Achs †　　Attila Kiss‡

Abstract

In this paper we define the fuzzy Datalog programs as sets of Horn-formulae with degrees and give their meaning by defining the deterministic and nondeterministic semantics. In the second part of the paper we show a possible extension of fDATALOG on fuzzy data.

1 Introduction

In knowledge-base systems there are given some facts representing certain knowledge and some rules which in general mean that certain kinds of information imply other kinds of information. In classical deductive database theory ([CGT], [U]) the Datalog-like data model is widely spread. Its most general type allows the use of both function symbols and negation. The meaning of a Datalog-like program is the least (if it exists) or a minimal model which contains the facts and satisfies the rules. This model is generally computed by a fixpoint algorithm.

The aim of this paper, which is partially a further development of [AK] and [K], is to give a possible extension of Datalog-like languages to fuzzy relational databases using lower bounds of degrees of uncertainty in facts and rules. We give a method for fixpoint queries. We show that this fixpoint is minimal under certain conditions.

We define the deterministic and nondeterministic semantics of fDATALOG and give a possible extension on fuzzy data.

2 The Concept of Fuzzy Datalog Program

To define the idea of fuzzy Datalog program (fDATALOG) we need some basic concepts.

A *term* is a variable, a constant or a complex term of the form $f(t_1, \ldots, t_n)$, where f is a function symbol and $t_1, \ldots, t_n$ are terms. An *atom* is a formula of the

*This research was supported by the Hungarian Foundation for Scientific Research Grant (OTKA) 2149 and OTKA 4281°

†Janus Pannonius University, Pollack Mihály College, Pécs, Boszorkány u. 2, Hungary

‡Eötvös Lóránd University, Budapest, Múzeum krt. 6-8., Hungary E-mail: kiss@tomx.elte.hu

form $p(\underline{t})$, where p is a predicate symbol of a finite arity (say n) and $\underline{t}$ is a sequence of terms of length n (arguments). A *literal* is either an atom (a positive literal) or the negation of an atom (a negative literal).

A term, atom, literal is *ground* if it is free of variables.

An *implication operator* is a mapping of the form

$$I(x,y) = \begin{cases} 1 & \text{if } x \le y \\ f(x,y) & \text{otherwise} \end{cases}$$

where $x, y \in [0,1]$ and $0 \le f(x,y) \le 1$.

Let D be a set. The *fuzzy set* F over D is a function $F : D \to [0,1]$. Let $\mathcal{F}(D)$ denote the set of all fuzzy sets over D. So $F \in \mathcal{F}(D)$.

$$F \cup G(d) \stackrel{\text{def}}{=} \max(F(d), G(d))$$

$$F \cap G(d) \stackrel{\text{def}}{=} \min(F(d), G(d)).$$

We can define an ordering relation: $F \le G$ iff $F(d) \le G(d)$, for $d \in D$.

The support of fuzzy set F is a classical set

$$\text{Supp}(F) = \{d | F(d) \neq 0\}.$$

We can see that $(\mathcal{F}(D), \le)$ is a complete lattice. The top element of the lattice is $U : D \to [0,1] : U(d) = 1$, for $d \in D$. The bottom element is: $\emptyset : D \to [0,1] : \emptyset(d) = 0$, for $d \in D$.

Fuzzy sets are frequently denoted in the following way:

$$F = \bigcup_{d \in D} (d, \alpha_d);$$

where $(d, \alpha_d) \in D \times [0,1]$.

In general the (d, α_d) pairs where $\alpha_d = 0$ are omitted from F, and sometimes $\text{Supp}(F)$ in enlarged with $(d, 0)$ pairs, where $d \in D$ but $d \notin \text{Supp}(F)$.

Below we will define the fuzzy Datalog language which is a possible extension of Datalog, using lower bounds of degrees of uncertainty in facts and rules. In this language the rules are completed with an implication operator and with a level. We allow for each formula to use any implication operator from a given set. Thus we fix a set of implication operator. We can infer the level of a rule's head from the level of the body and the level of the rule and the implication operator of the rule.

Definition 1 An fDATALOG rule is a triplet $(r; I; \beta)$, where r is a formula of the form

$$Q \leftarrow Q_1, \ldots, Q_n \qquad (n \ge 0)$$

where Q is an atom (the head of the rule), $Q_1, \ldots, Q_n$ are literals (the body of the rule); I is an implication operator and $\beta \in (0,1]$ (the level of the rule).

An fDATALOG rule is safe if

- All variables which occur in the head also occur in the body;
- All variables occuring in a negative literal also occur in a positive literal.

An fDATALOG program is a finite set of safe fDATALOG rules. Let A be a ground atom. The rules of the form $(A \leftarrow; I; \beta)$ are called facts.

The *Herbrand universe* of a program P (denoted by H_p) is the set of all possible ground terms constructed by using constants and function symbols occuring in P. The *Herbrand base* of P (B_p) is the set of all possible ground atoms whose predicate symbols occur in P and whose arguments are elements of H_p. A *ground instance* of a rule $(r; I; \beta)$ in P is a rule obtained from r by replacing every variable x in r by $\Phi(x)$ where Φ is a mapping from all variables occurring in r to H_p. The set of all ground instances of $(r; I; \beta)$ are denoted by $(ground(r); I; \beta)$. The ground instance of P is

$$ground(P) = \cup_{(r;I;\beta)\in P}(ground(r); I; \beta).$$

Definition 2 An interpretation of a program P, denoted by N_p, is a fuzzy set of B_p:

$$N_p \in \mathcal{F}(B_p), \text{ that is } N_p = \bigcup_{A \in B_p} (A, \alpha_A).$$

Let for ground atoms $A_1, \ldots, A_n$ $\alpha_{A_1 \wedge \ldots \wedge A_n}$ and $\alpha_{\neg A}$ be defined in the following way:

$$\alpha_{A_1 \wedge \ldots \wedge A_n} \overset{\text{def}}{=} \min(\alpha_{A_1}, \ldots, \alpha_{A_n}),$$

$$\alpha_{\neg A} \overset{\text{def}}{=} 1 - \alpha_A.$$

Definition 3 An interpretation is a model of P if for each $(ground(r); I; \beta) \in \text{ground}(P)$, $ground(r) = A \leftarrow A_1, \ldots, A_n$

$$I(\alpha_{A_1 \wedge \ldots \wedge A_n}, \alpha_A) \geq \beta$$

A model M is the least model if for any model $N, M \leq N$. A model M is minimal if there is no model $N \neq M$ such that $N \leq M$.

To be short we sometime denote $\alpha_{A_1 \wedge \ldots \wedge A_n}$ by α_{body} and α_A by α_{head}.

3 The Semantics of Fuzzy DATALOG

We will define two kinds of consequence transformations. Depending on these transformations we can define two semantics for fDATALOG. In this chapter we will show that the two semantics are the same in the case of positive programs, but they are different when the program has negative literals.

Definition 4 The consequence transformations $DT_p : \mathcal{F}(B_p) \to \mathcal{F}(B_p)$ and $NT_p : \mathcal{F}(B_p) \to \mathcal{F}(B_p)$ are defined as

$DT_p(X) = \{\cup\{(A, \alpha_A)\} | (A \leftarrow A_1, \ldots, A_n; I; \beta) \in ground(P),$

$(|A_i|, \alpha_{A_i}) \in X$ for each $1 \le i \le n, \alpha_A = \max(0, \min\{\gamma | I(\alpha_{\mathbf{body}}, \gamma) \ge \beta\})\} \cup X$

and

$NT_p(X) = \{(A, \alpha_A)\} | \exists (A \leftarrow A_1, \dots, A_n; I; \beta) \in \mathit{ground}(P),$

$(|A_i|, \alpha_{A_i}) \in X$ for each $1 \le i \le n, \alpha_A = \max(0, \min\{\gamma | I(\alpha_{\mathbf{body}}, \gamma) \ge \beta\})\} \cup X,$

$|A|$ denotes $p(\underline{c})$ if either $A = p(\underline{c})$ or $A = \neg p(\underline{c})$ where p is a predicate symbol with arity k and $\underline{c}$ is a list of k ground terms.

Note: $NT_p(X)$ has at most one more element than X while $DT_p(X)$ may have many new elements.

For any $T : \mathcal{F}(B_p) \to \mathcal{F}(B_p)$ transformation let

$$T_0 = \{\cup\{(A, \alpha_A)\} | (A \leftarrow; I; \beta) \in \mathit{ground}(P), \alpha_A = \max(0, \min\{\gamma | I(1, \gamma) \ge \beta\})\} \cup$$

$\{(A, 0) | \exists (B \leftarrow \dots \neg A \dots; I; \beta) \in \mathit{ground}(P)\}$ and let

$$T_1 = T(T_0)$$

$$\vdots$$

$$T_n = T(T_{n-1})$$

$$\vdots$$

$$T_q = \text{ least upper bound } \{T_\gamma | \gamma < \delta\} \text{ if } \delta \text{ is a limit ordinal.}$$

Proposition 1 Both DT_p and NT_p have a fixpoint, i.e., there exists $X \in \mathcal{F}(B_p)$ and $Y \in \mathcal{F}(B_p) : DT_p(X) = X$ and $NT_p(Y) = Y$.
If P is positive, then $X = Y$ and this is the least fixpoint. (That is for any $Z = T(Z) : X \le Z$.)

Proof: As [CGT] and [GS] show, if T is an inflationary transformation over a complete lattice L, then T has a fixpoint. (T is inflationary if $X \le T(X)$ for every $X \in L$). If T is monotone ($T(X) \le T(Y)$ if $X \le Y$), then T has a least fixpoint (see in [L]).

Since DT_p and NT_p are inflationary and $\mathcal{F}(B_p)$ is a complete lattice, thus they have an inflationary fixpoint.

If P is positive, then $DT_p = NT_p$ and this is monotone, which proves the proposition. □

We denote the fixpoints of the transformations by $lfp(DT_p)$ and $lfp(NT_p)$.

We show, that these fixpoints are models of P, so we can define the meaning of programs by these fixpoints.

Theorem 1 $lfp(DT_p)$ and $lfp(NT_p)$ are models of P.

Proof:
For $T = DT_p$ and $T = NT_p$ in *ground*(P) there are rules in the following forms:

a. $(A \leftarrow; I; \beta)$.
b. $(A \leftarrow A_1, \ldots, A_n; I; \beta)$; $(A, \alpha_A) \in \text{lfp}(T)$ and $(|A_i|, \alpha A_i) \in \text{lfp}(T), 1 \leq i \leq n$.
c. $(A \leftarrow A_1, \ldots, A_n; I; \beta); \exists i : (|A_i|, \alpha A_i) \notin \text{lfp}(T)$.

It was shown in [AK] that in all of these cases $I(\alpha_{\text{body}}, \alpha_A) \geq \beta$.

This model for function- and negation-free fDATALOG is the least model of P and this fixpoint can be reached in finite steps as given in [AK]. □

Now we can define the semantics of fDATALOG programs.

Definition 5 We define $\text{lfp}(DT_p)$ to be the deterministic semantics and $\text{lfp}(NT_p)$ to be the nondeterministic semantics of fDATALOG programs.

The next statement is obvious by Proposition 1.:

Proposition 2 For function- and negation-free fDATALOG, the two semantics are the same.

We can choose many kinds of implication operators, but we will use only four of those which were discussed in [AK]. They are:

$$I_1(x,y) = \begin{cases} 1 & \text{if } x \leq y \\ y & \text{otherwise} \end{cases} \qquad I_2(x,y) = \begin{cases} 1 & \text{if } x \leq y \\ 1-(x-y) & \text{otherwise} \end{cases}$$

$$I_3(x,y) = \begin{cases} 1 & \text{if } x \leq y \\ y/x & \text{otherwise} \end{cases} \qquad I_4(x,y) = \begin{cases} 1 & \text{if } x \leq y \\ 0 & \text{otherwise} \end{cases}$$

As the next example shows, if the program has any negation, the two semantics are different.

Example 1

1. $r(a) \leftarrow; I_1; 0.8$
2. $p(x) \leftarrow r(x), \neg q(x); I_1; 0.6$
3. $q(x) \leftarrow r(x); I_1; 0.5$
4. $p(x) \leftarrow q(x); I_1; 0.8$

Then $\text{lfp}(DT_p) = \{(r(a), 0.8); (p(a), 0.6; (q(a), 0.5)\}$.

In nondeterministic evaluation we can get different solutions, depending on the order of applied rules.

If the order of rules is 1., 2., 3., 4. then $(\text{lfp}NT_p) = \{(r(a), 0.8); (p(a), 0.6; (q(a), 0.5)\}$, but if the evaluating order is 1., 3., 2., 4. then the $\text{lfp}(NT_p) = \{(r(a), 0.8); (p(a), 0.5); (q(a), 0.5)\}$.

□

The set $\mathrm{lfp}(DT_p)$ is not always a minimal model as shown in the example above. So in applying the deterministic semantics it is not certain that the obtained fixpoint is minimal. In the nondeterministic case however it is minimal under certain conditions. This condition is the stratification. The stratification gives an evaluating sequence in which the negative literals are evaluated at first.

To stratify a program, it is necessary to define the concept of dependency graph. This is a directed graph, whose nodes are the predicates of P. There is an arc from predicate p to predicate q if there is a rule whose body contains p or $\neg p$ and whose head predicate is q.

A program is recursive, if its dependency graph has one or more cycles.

A program is stratified if whenever there is a rule with head predicate p and a negated body literal $\neg q$, there is no path in the dependency graph from p to q.

The stratification of a program P is a partition of the predicate symbols of P into subsets $P_1, \ldots, P_n$ such that the following conditions are satisfied:

a. if $p \in P_i$ and $q \in P_j$ and there is an edge from q to p, then $i \geq j$

b. if $p \in P_i$ and $q \in P_j$ and there is a rule with the head p whose body contains $\neg q$, then $i > j$.

A stratification specifies an order of evaluation. First we evaluate the rules whose head-predicates are in P_1 then those ones whose head-predicates are in P_2 and so on. The sets $P_1, \ldots, P_n$ are called the strata of the stratification.

A program P is called stratified if and only if it admits a stratification.

There is a very simple method for finding a stratification for a stratified program P in [CGT], [U].

Let P be a stratified fDATALOG program with stratification $P_1, \ldots, P_n$. Let P_i^* denote the set of all rules of P corresponding to stratum P_i, that is the set of all rules whose head-predicate is in P_i.

Let

$$L_1 = \mathrm{lfp}(NT_{P_1^*})$$

where the starting point of the computation is T_0 defined earlier.

$$L_2 = \mathrm{lfp}(NT_{P_2^*})$$

where the starting point of the computing is L_1,

$$\ldots$$

$$L_n = \mathrm{lfp}(NT_{P_n^*})$$

where the starting point is L_{n-1}.

In other words we first compute the least fixpoint L_1 corresponding to the first stratum of P. Once we computed this fixpoint we can take a step to the next strata.

Note:

$$\mathrm{lfp}(NT_{P_i^*}) = \mathrm{lfp}(DT_{P_i^*}).$$

We will show by induction that L_n is a minimal model of **P**. For this purpose we need the next lemma.

Lemma 1 Let P be an fDATALOG program such that for each negated predicate in a rule body there is not any rule whose head-predicate would be the same, but this predicate can occur among the facts. Then P has a least model:

$$L = \text{lfp}(NT_p)(= \text{lfp}(DT_p)).$$

Proof:

Let p be a negated predicate in a rule body of P. As there is not any new rule for p, therefore the degree of p will never change during the computation. For such P NT_p (or DT_p) is monotone and therefore $\text{lfp}(NT_p)$ is the least model of P.

□

According to this lemma, L_1 is the least fixpoint of P_1^*. Generally L_i is the least fixpoint of P_i^*, because due to the stratification of P, all negative literals of stratum i correspond to predicates of lower strata, so there is not any rule in P_i^* whose head-predicate would be this one.

From this we get the following theorem:

Theorem 2 If P is a stratified fDATALOG program then L_n is a minimal fixpoint of P.

Theorem 3 For stratified fDATALOG program P, there is an evaluation sequence, in which $\text{lfp}(NT_p)$ is a minimal model of P.

Proof:

The above construction gives this sequence. In this sequence $L_n = \text{lfp}(NT_p)$.

□

4 Connection Between fDATALOG and Fuzzy Relations

The ordinary Datalog maybe interpreted by relations such, that to every predicate p with arity k corresponds a relation P with arity k. The rows of P are those for which the predicate p is true.

Similarly it is possible to associate relations with fDATALOG predicates. The problem is that in the literature there are different definitions of fuzzy relations. Below we will deal with two kinds of fuzzy relations.

Definition 6 A first type fuzzy relation R in $D_1, \ldots, D_n$ is characterised by an n-variate membership function:

$$\mu_R : D_1 \times \ldots \times D_n \to [0, 1].$$

In other words a first type fuzzy relation is a fuzzy subset of the Cartesian Product of $D_1, \ldots, D_n$.

We will denote this relation by:

$$(R(D_1, \ldots, D_n), \mu_R).$$

Example 2 The relation $(F(X,Y), \mu_F)$ is a first type fuzzy relation, where F denotes the friends relation:

F:

X	Y	μ_F
John	Tom	0.1
Jim	Bob	0.6

□

It is obvious that we can relate a first type fuzzy relation to each predicate of an fDATALOG program P so that

$$t = (a_1, \ldots, a_n) \in (R(D_1, \ldots, D_n) \text{ if and only if}$$

$$(r(a_1, \ldots, a_n), \mu_R(t)) \in \text{ lfp } (DT_p) \text{ or lfp } (NT_p).$$

The first type fuzzy relations show the closeness of the connection among crisp data. However, sometimes we need to use fuzzy data. So we will define the second type fuzzy relation and will give a possible extension of fDATALOG on these relations.

Definition 7 Let $D_1, \ldots, D_n$ be n universal sets and $\mathcal{F}(D_1), \ldots, \mathcal{F}(D_n)$ be all their fuzzy sets. Then a second type fuzzy relation R is defined by an n-variate membership function:

$$\mu_R : \mathcal{F}(D_1) \times \ldots \times \mathcal{F}(D_n) \to [0,1].$$

Example 3 Let R be the following:

R:

Name	Age	Salary	μ_R
John	31	$\{0.8/3000, 0.7/3500\}$	0.7
Tom	middle aged	3300	0.8
Ann	young	$\{0.6/2000, 0.8/2500\}$	0.9

R is a second type fuzzy relation. □

5 Extension of fDATALOG for Fuzzy Data

We want to extend the fDATALOG programs so that the predicates of the programs can be related to second type fuzzy relations. Therefore we allow that the constants be any fuzzy data and the variables can have any fuzzy value.

Formally an fDATALOG rule is the same as above. Evaluating an extended fDATALOG program, we would have difficulties with the unification of fuzzy data, therefore we will complete these rules with similarity predicates, so the unification will be crisp and the uncertainty is expressed by the similarity.

Definition 8 $\text{sim}_D : \mathcal{F}(D) \times \mathcal{F}(D) \to [0,1]$ is a similarity predicate if it is reflexive and symmetric, that is, if $\text{sim}_D(x,x) = 1$ and $\text{sim}_D(x,y) = \text{sim}_D(y,x)$.

A similarity matrix is a matrix corresponding to similarity predicate.

A similarity is transitive if

$$\mathrm{sim}_D(x,z) \geq \max_{y\in D}\{\min(\mathrm{sim}_D(x,y), \mathrm{sim}_D(y,z))\}.$$

Definition 9 An extended fDATALOG program is an fDATALOG program on fuzzy data completed with similarity matrices.

If we want to evaluate an extended fDATALOG program, we have to transform it to an fDATALOG program. For this purpose we will build the similarity predicates into the rules. In rewriting predicates there are the following rules:

a. p is the head predicate of a fact

$$p(a_1,\ldots,a_n) \leftarrow; I; \beta.$$

Instead of this rule, we get:

$$p(x_1,\ldots,x_n) \leftarrow p(a_1,\ldots,a_n),\ \mathrm{sim}_{D_{x_1}}(a_1,x_1),\ldots,\ \mathrm{sim}_{D_{x_n}}(a_n,x_n); I; \beta,$$

where $\mathrm{sim}_{D_{x_i}}$ denotes the similarity predicate on the domain of x_i.

b. x is a variable in the rule

$$Q \leftarrow Q_1,\ldots,Q_m; I; \beta.$$

Suppose that x is in the predicates $p, p_{i_1},\ldots,p_{i_k}$ and there is no another occurence of x in this rule:

$$p(\ldots,x,\ldots), p_{i_1}(\ldots,x,\ldots), p_{i_k}(\ldots,x,\ldots).$$

Instead of this sequence we can write:

$$p(\ldots,x,\ldots), p_{i_1}(\ldots,x_1,\ldots),\ \mathrm{sim}_{D_x}(x_1,x), p_{i_2}(\ldots,x_2,\ldots),$$
$$\mathrm{sim}_{D_x}(x_2,x),\ \mathrm{sim}_{D_x}(x_2,x_1),\ldots,$$
$$p_{i_k}(\ldots,x_k,\ldots),\ \mathrm{sim}_{D_x}(x_k,x),\ \mathrm{sim}_{D_x}(x_k,x_1),\ldots,\ \mathrm{sim}_{D_x}(x_k,x_{k-1})$$

independently of whether p is in the head or in the body and independently of the order of the predicates (because of symmetry of sim).

Then the head of the new and of the original rule will be the same.

c. x is a repeated variable in predicate p:

$$p(\ldots,x,\ldots,x,\ldots,x,\ldots) - x \text{ occurs } k+1 \text{ times}.$$

Instead of this we get:

$$p(\ldots,x,\ldots,x_1,\ldots,x_k,\ldots)\ \mathrm{sim}_{D_x}(x_1,x),\ \mathrm{sim}_{D_x}(x_2,x),\ldots,$$
$$\mathrm{sim}_{D_x}(x_k,x),\ \mathrm{sim}_{D_x}(x_1,x_2),\ldots,\ \mathrm{sim}_{D_x}(x_1,x_k),\ldots,\ \mathrm{sim}_{D_x}(x_{k-1},x_k).$$

In this case the head of the new rule is the same if p is in the body and it is of the form $p(\ldots,x,\ldots,x_1,\ldots,x_k,\ldots)$ if p is the head predicate.

Proposition 3 Completing the rewriting of rules we get an ordinary fDATALOG program.

This program can be evaluated by deterministic or nondeterministic semantics.

Proposition 4 If sim_{D_x} is transitive then

$$\mathrm{sim}_{D_x}(x_1,x),\ \mathrm{sim}_{D_x}(x_2,x),\ldots,\ \mathrm{sim}_{D_x}(x_k,x),\ \mathrm{sim}_{D_x}(x_2,x_1),\ldots,$$
$$\mathrm{sim}_{D_x}(x_k,x_1),\ldots,\ \mathrm{sim}_{D_k}(x_k,x_{k-1})$$

can be simplified to:

$$\mathrm{sim}_{D_x}(x_1,x),\ \mathrm{sim}_{D_x}(x_2,x),\ldots,\ \mathrm{sim}_{D_x}(x_k,x).$$

Proof: We will prove that $\mathrm{sim}_{D_x}(x_2,x_1)$ can be ignored. The proof is similar for other cases. Because of transitivity

$$\mathrm{sim}_{D_x}(x_2,x_1) \geq \min(\ \mathrm{sim}_{D_x}(x_2,x),\ \mathrm{sim}_{D_x}(x,x_1)).$$

As in case of the rule $A \leftarrow A_1,\ldots,A_n$ $\alpha_{\text{body}} = \min(\alpha_{A_1},\ldots,\alpha_{A_n})$ so it is possible to leave out $\mathrm{sim}_{D_x}(x_2,x_1)$. □

We give an algorithm of rewriting an extended fDATALOG program P.

Algorithm:

```
Procedure rewriting
    facts := { the set of facts of P}
    rules := { the set of rules of P} - facts
    C := { the set of all possible constants }
                    (that is the union of domains)
    V := { the set of variables of P}

    while not_empty (facts) do
        fact := select (facts)
        arglist := { the list of arguments of the fact }
        i := 1
        body := ∅
        for a ∈ C do
            while a ∈ arglist do
                change (a, x_i)
                body := body ∧ sim_{D_a}(x_i, a)
                i := i + 1
            endwhile
        endfor
        facts := facts - fact
    endwhile
```

```
while not_empty (rules) do
        rule := select (rules)
        variable_lsit := {the list of variables of the rule }
        body := { the body of the rule }
        for x ∈ V do
                if x ∈ variable_list then
                        rest_list := leave_out (x, variable_list)
                        i := 1
                        while a∈ rest_list do
                                change (x, x_i)
                                body := body ∧ sim_{D_x}(x_i, x)
                                if non_transitive (sim_{D_x}) then
                                        for j :=1 to i − 1 do
                                                body := body ∧ sim_{D_x}(x_j, x_i)
                                        endfor
                                endif
                                i := i + 1
                        endwhile
                endif
        endfor
        rules:= rules - rule
endwhile
endprocedure
```

Example 4

$$p(a) \leftarrow; I_1; 0.7$$

$$r(b) \leftarrow; I_1; 0.8$$

$$q(x) \leftarrow p(x), r(x); I_1; 0.9$$

sim	a	b	c
a	1	0.8	0.1
b	0.8	1	0.7
c	0.1	0.7	1

The rewritten rules:

$$p(a) \leftarrow; I_1; 0.7$$

$$r(b) \leftarrow; I_1; 0.8$$

$$\text{sim}(a, a) \leftarrow; I_1; 1$$

$$\vdots$$

$$p(x) \leftarrow p(a),\ \text{sim}(x, a); I_1; 0.7$$

$$r(x) \leftarrow r(b),\ \text{sim}x, b); I_1; 0.8$$

$$q(x) \leftarrow p(x_1),\ \mathrm{sim}(x_1,x), r(x_2),\ \mathrm{sim}(x_2,x),\ \mathrm{sim}(x_2,x_1); I_1; 0.9$$

It is simpler to write the solution in matrix-form, so
$\mathrm{lfp}(NT_p) = \mathrm{lfp}(DT_p)$:

p:	a	b	c
	0.7	0.7	0.1

r:	a	b	c
	0.8	0.8	0.7

q:	a	b	c
	0.7	0.7	0.7

and the similarity matrix is the original. □

Example 5

$$p(a,b) \leftarrow; I_1; 0.9$$

$$p(c,d) \leftarrow; I_1; 0.8$$

$$q(x,y) \leftarrow p(x,y); I_1; 0.7$$

$$q(x,y) \leftarrow p(x,z), q(z,y); I_1; 0.8$$

sim	a	b	c	d	e
a	1	0	0.1	0.2	0.8
b	0	1	0.9	0.1	0
c	0.1	0.9	1	0.2	0
d	0.2	0.1	0.2	1	0.1
e	0.8	0	0	0.1	1

The rewritten rules (without facts and similarity matrix):

$$p(x,y) \leftarrow p(a,b),\ \mathrm{sim}(x,a),\ \mathrm{sim}(y,b); I_1; 0.9$$

$$p(x,y) \leftarrow p(c,d),\ \mathrm{sim}(x,c),\ \mathrm{sim}(y,b); I_1; 0.8$$

$$q(x,y) \leftarrow p(x_1,y_1),\ \mathrm{sim}(x_1,x),\ \mathrm{sim}(y_1,y); I_1; 0.7$$

$$q(x,y) \leftarrow p(x_1,z_1), q(z_2,y_1)\ \mathrm{sim}(x_1,x),\ \mathrm{sim}(y_1,y),\ \mathrm{sim}(z_1,z_2); I_1; 0.8$$

The fixpoint is:

p:	a	b	c	d	e
a	0.1	0.9	0.9	0.1	0.1
b	0.2	0.1	0.2	0.8	0.1
c	0.2	0.1	0.2	0.8	0.1
d	0.2	0.2	0.2	0.2	0.1
e	0	0.8	0.8	0.1	0

q:	a	b	c	d	e
a	0.2	0.7	0.7	0.7	0.2
b	0.2	0.2	0.2	0.7	0.2
c	0.2	0.2	0.2	0.7	0.2
d	0.2	0.2	0.2	0.2	0.2
e	0.2	0.7	0.7	0.7	0.2

and the original similarty matrix. □

Example 6

$$n(a) \leftarrow; I_1; 0.9$$

$$s(x,x) \leftarrow n(x); I_1; 0.8$$

sim	a	b	c
a	1	0.7	0.1
b	0.7	1	0
c	0.1	0	1

The rewritten rules (without facts and similarity matrix):

$$n(x) \leftarrow n(a),\ \mathrm{sim}(x,a); I_1; 0.9$$

$$s(x,x_1) \leftarrow\ \mathrm{sim}(x_1,x), n(x_2),\ \mathrm{sim}(x_2,x),\ \mathrm{sim}(x_1,x_2); I_1; 0.8$$

The fixpoint is:

n :	a	b	c
	0.9	0.7	0.1

s :	a	b	c
a	0.8	0.7	0.1
b	0.7	0.7	0
c	0.1	0	0.1

6 Conclusion

In this paper we gave a possible extension of Datalog-like languages. We defined the deterministic and nondeterministic semantics of fDATALOG and using the similarity relations we gave a possible extension on fuzzy data.

References

[AK] Ágnes Achs, Attila Kiss: Fixpoint query in fuzzy Datalog, to appear in Annales Universitatis Scientiarum Budapestinensis Sectio Computatorica, Budapest

[CGT] S. Ceri G. Gottlob L. Tanca: Logic Programming and Databases, Springer-Verlag, Berlin, 1990

[GS] Yuri Gurevich, Saharon Shelah:Fixed-point extensions of first-order logic, IEEEE Symp. on FOCS (1985), 346-353.

[K] Attila Kiss: On the least models of fuzzy Datalog programs, International Conference on Informaion Processing and Management of Uncertainty in Knowledge-based system, Mallorca 465-471.

[L] J. W. Lloyd: Foundations of Logic Programming, Springer-Verlag, Berlin, 1987.

[LL] Deyi Li, Dongbo Liu: A Fuzzy PROLOG Database System, Researc Studies Press LTD., Taunton, Somerset, England, 1990.

[N] Vilém Novák: Fuzzy sets and their applications, Adam Hilger Bristol and Philadelphia, 1987.

[U] J.D. Ullman: Principles of database and knowledge-base systems, Computer Science Press, Rockville, 1988.

Received June, 1995

Acta Cybernetica 12 (1995) 167-180.

Some Remarks On Generating Armstrong And Inferring Functional Dependencies Relation*

János Demetrovics† Vu Duc Thi†

Abstract

The main purpose of this paper is to give some results concerning algorithms for generating Armstrong relation and inferring functional dependencies (FDs for short). Firstly, we present some algorithms for solving these two problems. In the second part of the paper some NP-complete problems related to generating Armstrong relation and inferring FDs are given.

Key Words and Phrases: relation, relational datamodel, functional dependency, relation scheme, generating Armstrong relation, dependency inference, minimal key, antikey.

1 Introduction

Problems that construct a relation r such that r is an Armstrong relation of a given relation scheme (generating Armstrong relation) and a relation scheme s such that FDs of s hold in a given relation (inferring FDs) have been applied for for database design, query optimization, and artificial intelligence. These problems have been investigated in a lot of papers [3,9,12,16,17,18].

In this paper we give some results related to generating Armstrong relation and inferring FDs. The paper is structured as follows. In Section 2, we present some characterizations of the Armstrong relation of a given relation scheme, and construct an algorithm for finding all minimal transversals of a given hypergraph. From these and the results, presented in [9], we construct algorithms for generating Armstrong relation and inferring FDs.

Section 3 gives some NP-complete problems related to generating Armstrong relation and inferring FDs.

Let us give some necessary definitions and results that are used in the next sections. The concepts given in this section can be found in [1,3,4,6,7,8,10,11,13,19].

Let $R = \{a_1, \ldots, a_n\}$ be a nonempty finite set of attributes. A functional dependency is a statement of the form $A \to B$, where $A, B \subseteq R$. The FD $A \to B$

*Research supported by Hungarian Foundation for Scientific Research Grant 2575.

†Computer and Automation Institute Hungarian Academy of Sciences P.O.Box 63, Budapest, Hungary, H-1502

holds in a relation $r = \{h_1, \ldots, h_m\}$ over R if $\forall h_i, h_j \in r$ we have $h_i(a) = h_j(a)$ for all $a \in A$ implies $h_i(b) = h_j(b)$ for all $b \in B$. We also say that r satisfies the FD $A \to B$.

Let F_r be a family of all FDs that hold in r. Then $F = F_r$ satisfies

(1) $A \to A \in F$,

(2) $(A \to B \in F, B \to C \in F) \Longrightarrow (A \to C \in F)$,

(3) $(A \to B \in F, A \subseteq C, D \subseteq B) \Longrightarrow (C \to D \in F)$,

(4) $(A \to B \in F, C \to D \in F) \Longrightarrow (A \cup C \to B \cup D \in F)$.

A family of FDs satisfying (1)-(4) is called an f-family (sometimes it is called the full family) over R.

Clearly, F_r is an f-family over R. It is known [1] that if F is an arbitrary f-family, then there is a relation r over R such that $F_r = F$.

Given a family F of FDs, there exists a unique minimal f-family F^+ that contains F. It can be seen that F^+ contains all FDs which can be derived from F by the rules (1)-(4).

A relation scheme s is a pair $< R, F >$, where R is a set of attributes, and F is a set of FDs over R. Denote $A^+ = \{a: A \to \{a\} \in F^+\}$. A^+ is called the closure of A over s. It is clear that $A \to B \in F^+$ iff $B \subseteq A^+$.

Clearly, if $s =< R, F >$ is a relation scheme, then there is a relation r over R such that $F_r = F^+$ (see, [1]). Such a relation is called an Armstrong relation of s.

Let r be a relation, $s =< R, F >$ be a relation scheme. Then A is a key of r (a key of s) if $A \to R \in F_r$ ($A \to R \in F^+$). A is a minimal key of $r(s)$ if A is a key of $r(s)$ and any proper subset of A is not a key of $r(s)$.

Denote $K_r(K_s)$ the set of all minimal keys of $r(s)$.

Clearly, K_r, K_s are Sperner systems over R, i.e. $A, B \in K_r$ implies $A \not\subseteq B$.

Let K be a Sperner system over R. We define the set of antikeys of K, denoted by K^{-1}, as follows:

$$K^{-1} = \{A \subset R : (B \in K) \Longrightarrow (B \not\subseteq A)\, and\, (A \subset C) \Longrightarrow (\exists B \in K)(B \subseteq C)\}.$$

It is easy to see that K^{-1} is also a Sperner system over R.

Let R be a nonempty finite set, $P(R)$ its power set, and $I \subseteq P(R)$, $R \in I$, and $A, B \in I \Longrightarrow A \cap B \in I$. I is called a meet-semilattice over R. Let $M \subseteq P(R)$. Denote $M^+ = \{\cap M' : M' \subseteq M\}$. We say that M is a generator of I if $M^+ = I$. Note that $R \in M^+$ but not in M, by convention it is the intersection of the empty collection of sets.

Denote $N = \{A \in I : A \neq \cap\{A' \in I : A \subset A'\}\}$.

It can be seen that N is the unique minimal generator of I.

2 Algorithms

It is known [3,9,17] that the worst-case time complexities of generating Armstrong relation and inferring FDs are exponential. In this section we present some characterizations of the Armstrong relation of a given relation scheme. An effective algorithm finding all minimal transversals of a given hypergraph is also given. These results and the results, presented in [9], are used to construct algorithms for generating Armstrong relation and inferring FDs.

Let $s =< R, F >$ be a relation scheme. A FD $A \to \{a\} \in F^+$ is called the primitive maximal dependency (PMD for short) of s if $a \notin A$ and for all $A' \subseteq A : A' \to \{a\} \in F^+$ implies $A = A'$.

Denote $T_a = \{A : A \to \{a\}$ is a PMD of $s\}$. It can be seen that $\{a\}, R \notin T_a$, and T_a is a Sperner system over R. It is possible that $T_a = \emptyset$.

Let $s =< R, F >$ be a relation scheme, $a \in R$. Set $K_a = \{A \subseteq R: A \to \{a\}, \nexists B: (B \to \{a\})(B \subset A)\}$. K_a is called the family of minimal sets of the attribute a.

Clearly, $R \notin K_a$, $\{a\} \in K_a$ and K_a is a Sperner system over R. It is easy to see that $K_a - \{a\} = T_a$.

Based on the results, presented in [9], we show some characterizations of the Armstrong relation of a given relation scheme.

Lemma 2.1 [9] Let F be an f-family over R, $a \in R$. Denote $L_F(A) = \{a \in R: (A, \{a\}) \in F\}$, $Z_F = \{A: L_F(A) = A\}$. Clearly, $R \in Z_F$, $A, B \in Z_F \Longrightarrow A \cap B \in Z_F$. Denote by N_F the minimal generator of Z_F. Set $M_a = \{A \in N_F: a \notin A, \nexists B \in N_F: a \notin B, A \subset B\}$. Then $M_a = MAX(F, a)$, where $MAX(F, a) = \{A \subseteq R : A$ is a nonempty maximal set such that $(A, \{a\}) \notin F\}$.

Let r be a relation over R. Clearly, F_r is an f-family over R. Denote $L_{F_r}(A) = \{a \in R : A \to \{a\} \in F_r\}$, $Z_{F_r} = \{A : L_{F_r}(A) = A\}$. Put

$E_r = \{E_{ij} : 1 \le i < j \le |r|\}$, where $E_{ij} = \{a \in R : h_i(a) = h_j(a)\}$. E_r is called the equality set of r.

From E_r we compute $N = \{A \in E_r : A \neq \cap\{A' \in E_r : A \subset A'\}\}$. It can be seen that N is the minimal generator of Z_{F_r}. Then for each $a \in R$ we have

$M_a = \{A \in N : a \notin A, \nexists B \in N : A \subset B\}$.

It can be seen that $M_a = \{A \in E_r : a \notin A, \nexists B \in E_r : A \subset B\}$.

It is known [5] that an arbitrary full family of FDs can be uniquely determined by its primitive maximal dependencies.

From the result, presented in [9] (see, Remark 2.9), and Lemma 2.1 we obtain $K_a^{-1} = M_a$ for all $a \in R$. Clearly, if K is a Sperner system, then K and K^{-1} are uniquely determined by each other. Consequently, the next proposition is clear

Proposition 2.2 Let s be a relation scheme, and r a relation over R. Then r is an Armstrong relation of s if and only if for every $a \in R$

$$K_a^{-1} = M_a.$$

Now we present the concept of hypergraph that is in [4].

Let R be a nonempty finite set and $P(R)$ its power set. The family $H = \{E_i: E_i \in P(R), i = 1, \ldots, m\}$ is called a hypergraph over R if $E_i \neq \emptyset$. (In [4] author requires that the union of $E_i s$ is R, in this paper we do not).

A hypergraph H is simple if $E_i \subset E_j$ implies $i = j$, i.e., H is a Sperner system over R.

The elements of R are called vertices, and the sets $E_1, \ldots, E_m$ are the edges of the hypergraph H.

It is easy to see that a simple graph is a simple hypergraph with $|E_i| = 2$.

Let $H = \{E_1, \ldots, E_m\}$ be a hypergraph over R. Set

$m(H) = \{E_i \in H: \nexists E_j \in H : E_j \subset E_i\}$.

It can be seen that $m(H)$ is a simple hypergraph, and the family H uniquely determines the family $m(H)$.

Let H be a hypergraph over R. A set $A \subseteq R$ is called a transversal of H (sometimes it is called a hitting set) if $E \in H$ implies $A \cap E \neq \emptyset$.

The family of all minimal transversals of H is called the transversal hypergraph of H, and denoted by $tr(H)$. Clearly, $tr(H)$ is a simple hypergraph.

Remark 2.3 Let K be a Sperner system over R. Based on the definitions of K^{-1} and $tr(K)$ we can see that $tr(K) = \{R - A : A \in K^{-1}\}$.

Denote $N_a = \{R - A : A \in M_a\}$. From Proposition 2.2 and Remark 2.3 we have

Proposition 2.4 Let r be a relation, and s a relation scheme over R. Then r is an Armstrong relation of s iff for all $a \in R$

$$tr(K_a) = N_a.$$

It is known [4] that if H, H' are two simple hypergraph over R, then $H = tr(H')$ iff $H' = tr(H)$. From this and Remark 2.3, we can see that if K is a Sperner system, then $tr(\{R - A : A \in K^{-1}\}) = K$. According to the definitions of the set of all antikeys, the family of all minimal transversals, and Proposition 2.2 we obtain

Proposition 2.5 Let r be a relation, and s a relation scheme over R. Then r is an Armstrong relation of s iff for all $a \in R$

$$N_a^{-1} = \{B : R - B \in K_a\}.$$

Clearly, from Proposition 2.4 we have

Proposition 2.6 Let r be a relation, and s a relation scheme over R. Then r is an Armstrong relation of s iff for all $a \in R$

$$K_a = tr(N_a).$$

It is obvious that $a \in R - A$, where $A \in M_a$. Clearly, $T_a = K_a - \{a\}$. Thus, from the definition of the transversal hypergraph we obtain $T_a = tr(\{(R-a) - A : A \in M_a\})$ for all $a \in R\,(*)$.

Let r be a relation over R. A FD $A \to \{a\} \in F_r$ is called the primitive maximal dependency of r if $a \notin A$ and for all $A' \subseteq A : A' \to \{a\} \in F_r$ implies $A = A'$.

Denote $V_a = \{A : A \to \{a\}$ is a PMD of $r\}$, and $N'_a = \{(R-a) - A : A \in M_a\}$. By (*) and according to the definitions of F_r, and F^+ we have

Proposition 2.7 Let r be a relation over R. Then for all $a \in R$, $V_a = tr({N'}_a)$.

Proposition 2.7 was independently discovered in [18].

In this paper, we consider the comparison of two attributes as an elementary step of algorithms. Thus, if we assume that subsets of R are represented as sorted lists of attributes, then a Boolean operation on two subsets of R requires at most $|R|$ elementary steps.

Now we construct an algorithm that finds all minimal transversals of a given hypergraph.

Algorithm 2.8 (Finding all minimal transversals).

Input: Let $H = \{E_1, \ldots, E_m\}$ be a hypergraph over R.

Output: $tr(H)$.

Step 1: Set $L_1 = \{\{a\} : a \in E_1\}$. It is obvious that $L_1 = tr(\{E_1\})$.

Step q+1 $(q < m)$:

Assume that $L_q = S_q \cup \{B_1, \ldots, B_{t_q}\}$, where $B_i \cap E_{q+1} = \emptyset, i = 1, \ldots, t_q$ and $S_q = \{A \in L_q : A \cap E_{q+1} \neq \emptyset\}$.

For each i $(i = 1, \ldots, t_q)$ construct the set $\{B_i \cup b : b \in E_{q+1}\}$. Denote them by $A^i_1, \ldots, A^i_{r_i}$ $(i = 1, \ldots, t_q)$. Let

$$L_{q+1} = S_q \cup \{A^i_p : A \in S_q \Longrightarrow A \not\subset A^i_p,\ 1 \leq i \leq t_q,\ 1 \leq p \leq r_i\}.$$

Set $tr(H) = L_m$.

Theorem 2.9 For every q $(1 \leq q \leq m)$, $L_q = tr(\{E_1, \ldots, E_q\})$, i.e., $L_m = tr(H)$.

Proof. We prove this theorem by induction. It is obvious that $L_1 = tr(\{E_1\})$. We have to show that $L_{q+1} = tr(\{E_1, \ldots, E_{q+1}\})$. For this using the inductive hypothesis $L_q = tr(\{E_1, \ldots, E_q\})$.

Firstly, assume that D is the minimal subset of R such that $D \cap E_t \neq \emptyset\,(t = 1, \ldots, q+1)$. By the inductive hypothesis, there is a $X \in L_q$ such that $X \subseteq D$.

If $X \in S_q$, then $X \cap E_t \neq \emptyset$ for all $t = 1, \ldots, q+1$. Because D is the minimal subset of R such that $E_t \cap D \neq \emptyset\,(t = 1, \ldots, q+1)$, we have $X = D$. Hence, $D \in S_q$ holds. Consequently, we obtain $D \in L_{q+1}$.

If $X \cap E_{q+1} = \emptyset$, then $X = B_i$ holds for some i in $\{1, \ldots, t_q\}$. By $D \cap E_{q+1} \neq \emptyset$ we have $B_i \subset D$. Thus, $(D - B_i) \cap E_{q+1} \neq \emptyset$ holds. According to the construction of L_{q+1}, we have $A_p^i \subseteq D$ for some p in $\{1, \ldots, r_i\}$. Clearly, $A_p^i \cap E_l \neq \emptyset$ for all $l = 1, \ldots, q+1$, i.e., A_p^i is a transversal of the family $\{E_1, \ldots, E_{q+1}\}$. By $D \in tr(\{E_1, \ldots, E_{q+1}\})$ we obtain $D = A_p^i$. Because D does not contain the elements of S_q, we have $D \in L_{q+1}$.

Conversely, assume that $D \in L_{q+1}$. If $D \in S_q$, then $D \cap E_p \neq \emptyset \, (p = 1, \ldots, q)$ and D is minimal for this property, and at the same time $D \cap E_{q+1} \neq \emptyset$. Consequently, we have $D \in tr(E_1, \ldots, E_{q+1})$.

Let $D \in L_{q+1} - S_q$. Clearly, there is an $A_p^i \, (1 \leq i \leq t_q$ and $1 \leq p \leq r_i)$ such that $D = A_p^i$. Our construction shows that $E_l \cap A_p^i \neq \emptyset$ for all $l = 1, \ldots, q+1$. By the construction of algorithm we obtain $A_p^i = B_i \cup \{b\}$ for some $b \in E_{q+1}$.

Suppose that C is a proper subset of A_p^i, and $C \in tr(\{E_1, \ldots, E_{q+1}\})$. Clearly, $b \in C$ holds. According to the definitions of the transversal and the family of all minimal transversals, C is a transversal of the collection $\{E_1, \ldots, E_q\}$. By the inductive hypothesis ($L_q = tr(\{E_1, \ldots, E_q\})$), if there is $A \in S_q$ such that $A \subseteq C$, then we have $A \subset A_p^i$. This contradicts $A \not\subset A_p^i$ for all $A \in S_q$. If there is $B_j \; (1 \leq j \leq t_q) \, B_j \cap E_{q+1} = \emptyset$ such that $B_j \subseteq C$, then $b \notin B_j$ and $B_j \subset B_i$. This conflicts with the fact that L_q is a simple hypergraph. Hence, $D \in tr(\{E_1, \ldots, E_{q+1}\})$ holds.

Thus, $L_{q+1} = tr(\{E_1, \ldots, E_{q+1}\})$. Hence, $L_m = tr(H)$ holds. The theorem is proved.

It can be seen that the hypergraph H uniquely determines the family $tr(H)$, and the determination of $tr(H)$ based on our algorithm does not depend on the order of $E_1, \ldots, E_m$.

Remark 2.10 Denote $L_q = S_q \cup \{B_1, \ldots, B_{t_q}\}$, and $l_q \, (1 \leq q \leq m-1)$ is the number of elements of L_q. It can be seen that the worst-case time complexity of our algorithm is

$$O(|R|^2 \sum_{q=0}^{m-1} t_q u_q),$$

where $l_0 = t_0 = 1$ and

$$u_q = \begin{cases} l_q - t_q & \text{if } l_q > t_q, \\ 1 & \text{if } l_q = t_q. \end{cases}$$

Clearly, in each step of our algorithm L_q is a simple hypergraph. It is known that the size of arbitrary simple hypergraph over R can not be greater than $C_n^{[n/2]}$, where $n = |R|$. $C_n^{[n/2]}$ is asymptotically equal to $2^{n+1/2}/(\pi \cdot n)^{1/2}$. From this, the worst-case time complexity of our algorithm can not be more than exponential in the number of attributes. In cases for which $l_q \leq l_m (q = 1, \ldots, m-1)$, it is easy to see that the time complexity of our algorithm is not greater than $O(|R|^2 |H| |tr(H)|^2)$. Thus, in these cases this algorithm finds $tr(H)$ in polynomial time in $|R|, |H|$ and

$|tr(H)|$. Obviously, if the number of elements of H is small, then this algorithm is very effective. It only requires polynomial time in $|R|$.

It can be seen that our algorithm is better than the algorithm, presented in [4], finding all minimal transversals.

We give the next example which illustrates our algorithm.

Example 2.11 Let $R = \{1,2,3,4,5,6\}$, and
$H = \{(1,2), (2,3,4), (2,4,5), (4,6)\}$.

From Algorithm 2.8 we obtain

$$L_1 = \{(1), (2)\};$$

$$L_2 = \{(1,3), (1,4), (2)\};$$

$$L_3 = \{(1,3,5), (1,4), (2)\};$$

$$L_4 = \{(2,6), (2,4), (1,3,5,6), (1,4)\}.$$

Clearly, $tr(H) = L_4$.

Now we give the algorithm, presented in [9], that finds K_a

Algorithm 2.12 [9] (Finding a minimal set of the attribute a).

Input: Let $s = < R, F >$ be a relation scheme, $A = \{a_1, \ldots, a_t\} \to \{a\}$.

Output: $A' \in K_a$.

Step 0: We set $L(0) = A$.

Step i+1: Set

$$L(i+1) = \begin{cases} L(i) - a_{i+1} & \text{if } L(i) - a_{i+1} \to \{a\}, \\ L(i) & \text{otherwise.} \end{cases}$$

Then set $A' = L(t)$.

Algorithm 2.13 [9] (Finding a family of all minimal sets of attribute a).

Input: Let $s = < R, F >$ be a relation scheme, $a \in R$.

Output: K_a.

Step 1: Set $L(1) = E_1 = \{a\}$.

Step i+1: If there are C and $A \to B$ such that $C \in L(i)$, $A \to B \in F$, $\forall E \in L(i) \implies E \not\subseteq A \cup (C - B)$, then by Algorithm 2.12 construct an E_{i+1}, where $E_{i+1} \subseteq A \cup (C - B)$, $E_{i+1} \in K_a$. We set $L(i+1) = L(i) \cup E_{i+1}$. In the converse case we set $K_a = L(i)$.

It is shown [9] that there exists a natural number n such that $K_a = L(n)$.
It can be seen that the worst-case time complexity of algorithm is
$O(|R||F||K_a|(|R| + |K_a|))$.
Thus, the time complexity of this algorithm is polynomial in $|R|, |F|$, and $|K_a|$.
Clearly, if the number of elements of K_a for a relation scheme $s =< R, F >$ is polynomial in the size of s, then this algorithm is effective. Especially, when $|K_a|$ is small.

Based on Proposition 2.4, Algorithms 2.8 and 2.13 we construct the next algorithm.

Algorithm 2.14 (Generating Armstrong relation).
Input: Let $s =< R, F >$ be a relation scheme.
Output: A relation r such that $F_r = F^+$.
Step 1: For each $a \in R$ by Algorithm 2.13 we compute K_a, and from Algorithm 2.8 find $tr(K_a)$.
Step 2: $N = \bigcup_{a \in R} tr(K_a)$
Step 3: Denote elements of N by $A_1, \ldots, A_t$ construct a relation
$R = \{h_0, h_1, \ldots, h_t\}$ as follows:
For all $a \in R$, $h_0(a) = 0$, $\forall i = 1, \ldots, t$

$$h_i(a) = \begin{cases} i & \text{if } a \in A_i \\ 0 & \text{otherwise} \end{cases}$$

It is known [16] that if $s =< R, F >$ is a relation scheme. Denote $Z_s = \{A: A^+ = A\}$, and N_s is a minimal generator of Z_s. Then

$$N_s = \bigcup_{a \in R} MAX(F^+, a)$$

where

$$MAX(F^+, a) = \{A \subseteq R: A \to \{a\} \notin F^+, A \subset B \Longrightarrow B \to \{a\} \in F^+\}.$$

From this and the definitions of M_a, and N_a of the relation r we have $tr(K_a) = N_a$ for all $a \in R$. Consequently, by Proposition 2.4 we obtain $F_r = F^+$.

The estimation and the effectiveness of this algorithm are analogous to the algorithm, presented in [9] (see, Remark 2.12 in [9]), so its proof will be omitted.

Now we give the algorithm finding all antikeys, presented in [20].
Let $K = \{B_1, \ldots, B_m\}$ be a Sperner system over R.
For each $q = 1, \ldots, m$ we construct $K_q = \{B_1, \ldots, B_q\}^{-1}$ by induction.
Set $K_1 = \{R - \{a\} : a \in B_1\}$. It is obvious that $K_1 = \{B_1\}^{-1}$.
By the inductive hypothesis we have constructed $K_q = \{B_1, \ldots, B_q\}^{-1}$ for $(q < m)$.

We assume that $K_q = F_q \cup \{X_1, \ldots, X_{t_q}\}$, where $X_1, \ldots, X_{t_q}$ containing B_{q+1} and $F_q = \{A \in K_q : B_{q+1} \not\subseteq A\}$.

For all i $(i = 1, \ldots, t_q)$ construct the antikeys of $\{B_{q+1}\}$ on X_i in an analogous way as K_1. Denote them by $A_1^i, \ldots, A_{r_i}^i \quad (i = 1, \ldots, t_q)$. Let

$$K_{q+1} = F_q \cup \{A_p^i : A \in F_q \Longrightarrow A_p^i \not\subset A,\ 1 \leq i \leq t_q,\ 1 \leq p \leq r_i\}.$$

Set $K^{-1} = K_m$.

Denote $K_q = F_q \cup \{X_1, \ldots, X_{t_q}\}$ and $l_q \quad (1 \leq q \leq m-1)$ is the number of elements of K_q.

Remark 2.15 [20] The time complexity of algorithm finding all antikeys is

$$O(|R|^2 \sum_{q=0}^{m-1} t_q u_q),$$

where

$$u_q = \begin{cases} l_q - t_q & \text{if } l_q > t_q \\ 1 & \text{if } l_q = t_q \end{cases}$$

According to Proposition 2.5 and the algorithm finding all antikeys we will construct the following algorithm.

Algorithm 2.16 (Inferring FDs).

Input: r be a relation over R.

Output: $s = < R, F >$ such that $F^+ = F_r$.

Step 1: From r compute the equality set E_r

Step 2: Set $N = \{A \in E_r : A \neq \cap\{B \in E_r : A \subset B\}\}$

Step 3: For each $a \in R$ find $M_a = \{A \in N_R : a \notin A,\ \nexists B \in N_R : a \notin B,\ A \subset B\}$. Compute $N_a = \{R - A : A \in M_a\}$.

Step 4: By the algorithm finding all antikeys, for each $a \in R$ construct N_a^{-1}.

Step 5: Construct $s = < R, F >$, where $F = \{R - B \to \{a\} : \forall a \in R,\ B \in N_a^{-1}, R - B \neq \{a\}\}$

By Proposition 2.5 we have $F_r = F^+$.

Remark 2.17 Clearly, for all $a \in R$ N_a is computed in polynomial time in the size of r. It can be seen that the complexity of Algorithm 2.16 is the complexity of step 4. By Remark 2.15, it is easy to see that the worst-case time complexity of Algorithm 2.16 is

$$O(n^2 \sum_{i=1}^{n} (\sum_{q=0}^{m_i - 1} t_{iq} u_{iq}))$$

where $R = \{a_1, \ldots, a_n\}$, $m_i = |N_{a_i}|$ and

$$u_{iq} = \begin{cases} l_{iq} - t_{iq} & \text{if } l_{iq} > t_{iq} \\ 1 & \text{if } l_{iq} = t_{iq} \end{cases}$$

Meaning of l_{iq}, t_{iq}, u_{iq} see Remark 2.15.

In cases for which $l_{iq} \leq l_{m_i} (\forall i, \forall q{:}\, 1 \leq q \leq m_i)$ the time complexity of our algorithm is $O(n^2 \sum_{i=1}^{n} |N_{a_i}||N_{a_i}{}^{-1}|^2)$. Thus, the complexity of Algorithm 2.16 is polynomial in $|R|$, $|N_{a_i}|$, $|N_{a_i}{}^{-1}|$. Clearly, in these cases if $|N_{a_i}{}^{-1}|$ is polynomial (Especially, it is small) in the size of r, then our algorithm is effective.

According to Proposition 2.6 and algorithm 2.8 we give the next algorithm for inferring FDs.

Algorithm 2.18 (Inferring FDs).

Input: r be a relation over R.

Output: $s =< R, F >$ such that $F^+ = F_r$.

Step 1: From r compute the set N_a for all $a \in R$.

Step 2: By Algorithm 2.8, construct $tr(N_a)$, for every $a \in R$.

Step 3: Construct $s =< R, F >$, where $F = \{A \to \{a\}{:}\, \forall a \in R, A \in tr(N_a), A \neq \{a\}\}$.

By Proposition 2.6 we have $F_r = F^+$.

The estimation of Algorithm 2.18 is analogous to Algorithm 2.16, so its proof will be omitted.

It can be seen that Algorithm 2.18 is similar to the algorithm inferring FDs, presented in [18]. However, it can be seen that Algorithm 2.8 is better than the algorithm, presented in [4], that is used in [18].

3 NP-complete Problems

In this section, we present some NP-complete problems related to PMDs, and the sets M_a. In Section 2, we show that these sets play important roles in generating Armstrong relation and inferring FDs.

Let $s =< R, F >$ be a relation scheme over R. Denote $L_a = \{A : A \to \{a\}, a \notin A\}$. It can be seen that L_a contains all PMDs concerning a, i.e., $T_a \subseteq L_a$.

Firstly, we introduce the following problem related to the set L_a.

Theorem 3.1 The following problem is NP-complete:

Let $s =< R, F >$ be a relation scheme over R, $a \in R$, and an integer m ($m \leq |R|$), decide whether there is an A such that $a \notin A$, $A \to \{a\}$, (i.e., $A \in L_a$), and $|A| \leq m$.

Proof. We nondeterministically choose a set A so that $|A| \leq m$, $a \notin A$, and decide whether $A \to \{a\}$ is an element of F^+. Clearly, by the polynomial time algorithm finding the closure (see [2]), our algorithm is nondeterministic polynomial. Thus, our problem lies in NP.

Now we shall show that our problem is NP-hard. It is known [15] that the problem deciding whether there exists a key having cardinality less than or equal to a given integer for relation scheme is NP-complete. Now we prove that this problem is polynomially reducible to our problem.

Let $s' =< P, F' >$ be a relation scheme over P. Now we construct the relation scheme $s =< R, F >$, as follows:

$R = P \cup a$, where $a \notin P$ and $F = F' \cup P \to \{a\}$.

It is obvious that s is constructed in polynomial time in the sizes of P and F'. Based on the construction of s and the definition of the minimal key we can see that if $A \in K_{s'}$, then $A \in K_s$. Conversely, if B is a minimal key of s, then by $R \to \{a\} \in F$ we have $a \notin B$. On the other hand, by the definition of the minimal key $B \in K_{s'}$. Thus, $K_{s'} = K_s$ holds. By $P \to \{a\} \in F$, and $a \notin P$, if $B \to \{a\}$ is a PMD of s, then $B \in K_s$. It can be seen that if $A \in K_{s'}$, then $A \to \{a\} \in F^+$. According to the definition of PMD, $A \to \{a\}$ is a PMD of s. Consequently, C is a key of s' if and only if $a \notin C$, and $C \to \{a\} \in F^+$. The theorem is proved.

Now we give the NP-complete problem concerning M_a, (see, Lemma 2.1).

Theorem 3.2 The following problem is NP-complete:

Let $s =< R, F >$ be a relation scheme over R, $a \in R$, and an integer $m\,(m \leq |R|)$, decide whether there is an A such that $a \notin A$, $A \to \{a\} \notin F^+$, and $m \leq |A|$.

Proof. By the proof of Theorem 3.1, it is clear that our problem lies in NP.

It is known [14] that the independent set problem is NP-complete :

Given integer m and a non-directed graph $G =< V, E >$, where V is the set of vertices and E is the set of edges. An independent set in G is a subset $A \subseteq V$ such that for all $a, b \in A$, the edge (a, b) is not in E. The independent set problem is deciding whether G contains an independent set A having cardinality greater than or equal to m.

We shall prove that the independent set problem is polynomially reducible to our problem.

Let $G =< V, E >$ be a non-directed graph, $m \leq |V|$. Set

$s' =< V, F' >$, where $F' = \{\{a_i, a_j\} \to V : (a_i, a_j) \in E\}$, and

$s =< R, F >$, where $R = V \cup \{a\}$, $a \notin V$, and $F = F' \cup V \to \{a\}$.

Clearly, s, s' are constructed in polynomial time in the size of G.

According to the definition of the set of edges, E is a simple hypergraph over V. From this, we can see that s' is in BCNF. Because E is the set of edges, and by the definition of the minimal key, we can see that if $(a_i, a_j) \in E$, then $\{a_i, a_j\}$ is a minimal key of s'. Conversely, if $B \in K_{s'}$, then there is an $\{a_i, a_j\}$ such that $\{a_i, a_j\} \subseteq B$. Because B is a minimal key, we have $\{a_i, a_j\} = B$. Hence, $K_{s'} = E$ holds.

Consequently, A is not a key of s' if and only if $\{a_i, a_j\} \notin A$ for all $(a_i, a_j) \in E$. Thus, A is not a key of s' if and only if A is an independent set of G.

On the other hand, by the proof of Theorem 3.1 C is a key of s' if and only if $C \to \{a\} \in F^+$, and $a \notin C$. Consequently, A is not a key of s' if and only if $a \notin A$, and $A \to \{a\} \notin F^+\}$.

Thus, A is an independent set of G if and only if A does not contain a, and $A \to \{a\} \notin F^+$. The theorem is proved.

Now we will show that Theorem 3.1 is still true for the relations.

Theorem 3.3 The following problem is NP-complete:

Let r be a relation over R, $a \in R$, and an integer $m\,(m \leq |R|)$, decide whether there is an A such that $a \notin A$, $A \to \{a\} \in F_r$, and $|A| \leq m$.N

Proof.

We nondeterministically choose a set A so that $|A| \leq m$, $a \notin A$, and decide whether $A \to \{a\} \in F_r$. Clearly, using the definition of the functional dependency, we can test in polynomial time that the functional dependency $A \to \{a\}$ holds or does not hold in r. It is obvious that our algorithm is nondeterministic polynomial. Thus, the problem lies in NP.

It is shown [14] that the the vertex cover problem is NP-complete:

Given integer m and a non-directed graph $G =< V, E >$, where V is the set of vertices and E is the set of edges, decide whether G has a vertex cover having cardinality not greater than m.

Let $G =< V, E >$ be a non-directed graph, $m \leq |V|$. Put $R = V \cup a$, where $a \notin V$.

Denote the elements of E by $E_1, \ldots, E_t$ construct a relation

$r = \{h_0, h_1, \ldots, h_t\}$, as follows:

For all $b \in R$, $h_0(b) = 0$, $\forall i = 1, \ldots, t$

$$h_i(b) = \begin{cases} i & \text{if } b \in E_i \text{ or } b = a \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, the set E is a Sperner system. From this and by the definition of N_a we can see that $N_a = \{\{a_i, a_j, a\} : (a_i, a_j) \in E\}$. Consequently, we obtain $N'_a = \{\{a_i, a_j\} : (a_i, a_j) \in E\}$. According to Proposition 2.7, $V_a = tr(\{\{a_i, a_j\} : (a_i, a_j) \in E\})$. On the other hand, by the definition of the vertex cover we can see that A is a vertex cover of G if and only if A does not contain a, and $A \to \{a\}$ is an element of F_r. The proof is complete.

Thus, for the relations Theorem 3.1 is still true. However, the next proposition shows that the problem, presented in Theorem 3.2, can be solved in polynomial time if the relation scheme is changed to the relation.

Proposition 3.4 Let r be a relation over R, $a \in R$, and an integer $m\,(m \leq |R|)$. Then the problem deciding whether there is an A such that $a \notin A$, $A \to \notin F_r$, and $m \leq |A|$ can be solved by a polynomial time algorithm.

Proof.

According to the difinitions of M_a and the antikey, and by Proposition 2.2 we can see that M_a is the family of all maximal sets A such that A doesn't contain a, and $A \to \{a\} \notin F_r$. Clearly, for every $a \in R$, we can compute the family M_a in polynomial time in the size of r.

Consequently, for relations, given an attribute a, and an integer m the problem deciding whether there is an A such that $a \notin A$, $A \to \{a\}$, and the cardinality of A is greater than or equal to m can be solved by a polynomial time algorithm. The proposition is proved.

References

[1] Armstrong W.W. Dependency Structures of Database Relationships. Information Processing 74, Holland Publ. Co. (1974) pp. 580-583.

[2] Beeri C., Bernstein P.A. Computational problems related to the design of normal form relational schemas. ACM Trans. on Database Syst. 4,1 (1979) pp. 30-59.

[3] Beeri C., Dowd M., Fagin R., Staman R. On the Structure of Armstrong relations for Functional Dependencies. J.ACM 31,1 (1984) pp. 30-46.

[4] Berge C. Hypergraphs : Combinatorics of Finite Sets (North - Holland, Amsterdam, 1989)

[5] Demetrovics J., Libkin L., Muchnik I.B. Functional dependencies and the semilattice of closed classes. Proceedings of MFDBS 87, Lecture Notes in Computer Science (1987) pp. 136-147.

[6] Demetrovics J., Thi V.D. Some results about functional dependencies. Acta Cybernetica 8,3 (1988) pp. 273-278.

[7] Demetrovics J., Thi V.D. Relations and minimal keys. Acta Cybernetica 8,3 (1988) pp. 279-285.

[8] Demetrovics J., Thi V.D. On Keys in the Relational Datamodel. Inform. Process. Cybern. EIK 24 (1988) 10, pp. 515-519.

[9] Demetrovics J., Thi V.D. Algorithms for generating Armstrong relation and inferring functional dependencies in the relational datamodel. Computers and Mathematics with Applications, Great Britain, 26, 4 (1993) pp. 43-55.

[10] Demetrovics J., Thi V.D. Some Problems concerning Keys for Relation Schemes and Relations in the Relational Datamodel. Information Processing Letters, North Holland 46, 4 (1993) pp. 179-183.

[11] Demetrovics J., Thi V.D. Some Computational Problems Related to the Functional Dependency in the Relational Datamodel. Acta Scientiarum Mathematicarum 57, 1-4 (1993) pp. 627-628.

[12] Demetrovics J., Thi V.D. Generating Armstrong relations for relation schemes and inferring functional dependencies from relations. International Journal on Information Theories and Applications, 1, 4 (1993) pp. 3-12.

[13] Demetrovics J., Thi V.D. Armstrong Relation, Functional Dependencies and Strong Dependencies. Comput. and AI, submitted for publication 1994.

[14] Garey M. R., Johnson D. S. Computers and Intractability : A Guide to the Theory of NP-Completeness. Bell Laboratories, W.H. Freeman and Company, San Francisco 1979.

[15] Lucchesi C.L., Osborn S.L. Candidate keys for relations. J. Comput. Syst. Scien. 17,2 (1978) pp. 270-279.

[16] Mannila H., Raiha K.J. Design by Example: An Application of Armstrong relations. J. Comput. Syst. Scien. 33 (1986) pp. 126-141.

[17] Mannila H., Raiha K.J. On the complexity of inferring functional dependencies. Discrete Applied Mathematics, North-Holland, 40 (1992) pp. 237-243.

[18] Mannila H., Raiha K.J. Algorithms for inferring functional dependencies from relations. Data and Knowledge Engineering, North Holland, 12 (1994) pp 83-99.

[19] Thi V.D. Investigation on Combinatorial Characterizations Related to Functional Dependency in the Relational Datamodel (in Hungarian). MTA-SZTAKI Tanulmanyok, Budapest, 191 (1986) pp. 1-157. Ph.D. Dissertation.

[20] Thi V.D. Minimal keys and Antikeys. Acta Cybernetica 7,4 (1986) pp. 361-371.

Received August, 1994

Acta Cybernetica 12 (1995) 181-207.

A Lattice View of Functional Dependencies in Incomplete Relations

Mark Levene *

Abstract

Functional Dependencies (or simply FDs) are by far the most common integrity constraint in the real world. When relations are incomplete and thus contain null values the problem of whether satisfaction is additive arises. Additivity is the property of the equivalence of the satisfaction of a set of functional dependencies (FDs), F, with the individual satisfaction of each member of F in an incomplete relation. It is well known that, in general, satisfaction of FDs is not additive. Previously we have shown that satisfaction is additive if and only if the set of FDs is monodependent. Thus monodependence of a set of FDs is a desirable property when relations may be incomplete. A set of FDs is monodependent if it satisfies both the intersection property and the split-freeness property. (The two defining properties of monodependent sets of FDs correspond to the two defining properties of conflict-free sets of multivalued data dependencies.)

We investigate the properties of the lattice $\mathcal{L}(F)$ of closed sets of a monodependent set of FDs F over a relation schema R. We show an interesting connection between monodependent sets of FDs and exchange and antiexchange lattices. In addition, we give a characterisation of the intersection property in terms of the existence of certain distributive sublattices of $\mathcal{L}(F)$. Assume that a set of FDs F satisfies the intersection property. We show that the cardinality of the family $\mathcal{M}(F)$ of meet-irreducible closed sets in $\mathcal{L}(F)$ is polynomial in the number of attributes associated with R; in general, this number is exponential. Thus an Armstrong relation for F having a polynomial number of tuples in the number of attributes associated with R can be generated. As a corollary we show that the prime attribute problem can be solved in polynomial time in the size of F; in general, the prime attribute problem is NP-complete. We also show that F satisfies the intersection property if and only if the cardinality of each element in $\mathcal{M}(F)$ is greater than or equal to the cardinality of the attribute set of R minus two. Using this result we are able to show that the superkey of cardinality k problem is still NP-complete when F is restricted to satisfy the intersection property. Finally, we show that separatory sets of FDs are monodependent.

*Department of Computer Science, University College London Gower Street, London, WC1E 6BT, U.K. email: mlevene@cs.ucl.ac.uk

1 Introduction

In order to handle incomplete information, Codd [CODD79] suggested the addition to the database domains of an unmarked null value, denoted by *unk*, whose meaning is "value at present exists but is unknown". We call relations, whose tuples may contain the null value *unk*, *incomplete relations*. The semantics of an incomplete relation r are defined in terms of the *possible worlds* relative to r. Each possible world relative to r is a *complete relation*, i.e. a relation without any occurrence of *unk*, emanating from a possible substitution of all the occurrences of *unk* in r by nonnull values in the underlying database domains.

Functional Dependencies (or simply FDs) are by far the most common integrity constraint in the real world [ULLM88, ATZE93, MANN92] and the notion of a key (derived from a given set of FDs) [CODD79] is fundamental to the relational model. Given a set of FDs F over a relation schema R and an incomplete relation r over R, it is therefore natural to say that r satisfies F if there is a complete relation s, in the set of possible worlds relative to r, such that s satisfies each of the FDs in F. This gives rise to the *additivity problem*, which is the problem of whether the statement that r satisfies F is equivalent to the statement that r satisfies each FD in a reduced cover G of F [LEVE94, LEVE95a] (cf. [ATZE93]); if these two statements are equivalent for a class of incomplete relations and a class of sets of FDs then we say that *satisfaction is additive* with respect to these classes. It is well known that, in general, satisfaction of FDs is not additive [ATZE86, LEVE94, LEVE95a]. If satisfaction is not additive, then a set of FDs F in this nonadditive class may be viewed as contradictory. Thus we consider the solution of the additivity problem to be an important prerequisite for any relational database system supporting FDs in the context of incomplete information, since otherwise semantic anomalies may arise.

In [LEVE94] we introduced the class of *monodependent* sets of FDs. A set of FDs F over a relation schema R is monodependent if the following two properties are satisfied. The first property, called the *intersection property*, informally states that for each attribute A in the attribute set associated with R, there is a unique nontrivial and reduced FD in the closure of F that functionally determines A. The second property, called the *split-freeness property*, informally states that there are no two nontrivial FDs in the closure of F such that the right-hand side of each of the two FDs splits the left-hand side of the other FD. The main result in [LEVE94] shows that satisfaction is additive with respect to the class of all incomplete relations and a class of sets of FDs **FC**, if and only if all the sets of FDs in **FC** are monodependent sets of FDs. Therefore, monodependence provides a solution to the additivity problem.

In [LEVE95b] we studied the impact on normalisation theory in relational databases of assuming that sets of FDs are monodependent, and in [LEVE95a] we extended the results in [LEVE94] to the wider class of sets of FDs and *unary inclusion dependencies* [COSM90].

It is well known that the family of all closed sets, with respect to a set of

FDs F, is a lattice partially ordered by set inclusion; we denote this lattice by $\mathcal{L}$(F) [DEME92, DEME93]. An in-depth investigation concerning the connection between the structure of a set of FDs and the type of lattice of closed sets it induces was carried out in [DEME92]. Herein we investigate the properties of $\mathcal{L}$(F) when F is monodependent.

We next briefly outline the main results of this paper. The set of *equivalence classes* of a set of FDs F over R is a partition of F such that two FDs are in the same equivalence class if and only if the closures of their left-hand sides are the same [MAIE80, MANN83]. Assume that F satisfies the intersection property. We then show that $\mathcal{L}$(F) is *exchange* [GRAT78] if and only if F satisfies the split-freeness property and the cardinality of all the nonempty equivalence classes of F is maximal. Correspondingly, we show that $\mathcal{L}$(F) is *antiexchange* [JAMI85] if and only if F satisfies the split-freeness property and the cardinality of all the nonempty equivalence classes of F is minimal, i.e. one. We conclude that the lattice of closed sets of a monodependent set of FDs is something in between an exchange and antiexchange lattice according to the cardinalities of its equivalence classes.

We also investigate some of the characteristics of the lattice $\mathcal{L}$(F) when the set of FDs F satisfies the intersection property but not necessarily the split-freeness property. We give a characterisation of the intersection property in terms of the existence of certain distributive sublattices of $\mathcal{L}$(F). We then present a polynomial time algorithm in the size of F to compute the set of *meet-irreducible* closed sets in $\mathcal{L}$(F), which we denote by $\mathcal{M}$(F) (see Definition 6.1). Let n be the cardinality of the attribute set of R. As a corollary of this algorithm we show that the cardinality of $\mathcal{M}$(F) is at most $\binom{n}{n-2}$; in general, this number is exponential in n. Thus an Armstrong relation having $\binom{n}{n-2} + 1$ tuples can be generated [MANN86]. As an additional corollary of this algorithm we show that testing whether an attribute is *prime* (see Definition 4.2) when F satisfies the intersection property can be done in polynomial time in the size of F; in general, testing whether an attribute is prime is NP-complete [LUCC78]. We also show that F satisfies the intersection property if and only if the cardinality of each element in $\mathcal{M}$(F) is greater than or equal to $n - 2$. Another well known problem, which is NP-complete in the general case, is the problem of deciding whether there exists a superkey for R of cardinality k or less [LUCC78, DEME88]. Utilsing this result we are able to show that this problem is still NP-complete when F satisfies the intersection property. Finally, we show that separatory sets of FDs are monodependent.

The layout of the rest of the paper is as follows. In Section 2 we formalise the notion of incomplete relations. In Section 3 we define the notion of a functional dependency being satisfied in an incomplete relation. In Section 4 we present the relevant properties of FDs which are utilised in the paper. In Section 5 we introduce monodependent sets of FDs and give some technical results, which are utilised in the following sections. In Section 6 we introduce the lattice-theoretic concepts that are used in the remaining sections. In Section 7 we give some negative results concerning the structure of $\mathcal{L}$(F) when F is monodependent. In Section 8 we investigate the connection between exchange and antiexchange lattices of closed sets and monodependent sets of FDs. In Section 9 we investigate some of the characteristics of lattices of closed sets of FDs that satisfy the intersection property. In Section 10 we show that separatory sets of FDs are monodependent. Finally, in Section 11 we give our concluding remarks.

2 Relations that model incomplete information

Herein we formalise the notion of an incomplete relation, which allows us to model incomplete information of the form "value at present exists but is unknown".

We use the notation |S| to denote the cardinality of a set S. If S is a subset of T we write S $\subseteq$ T and if S is a proper subset of T we write S $\subset$ T. Furthermore, S and T are *incomparable* if S $\not\subseteq$ T and T $\not\subseteq$ S. At times we denote the singleton {A} simply by A, and the union of two sets S and T, i.e. S $\cup$ T, simply by ST. The power set of a set S is denoted by $\mathcal{P}$(S).

Definition 2.1 (Relation schema and relation) A *relation schema* R is a finite set of attributes which we denote by schema(R); we denote the cardinality of schema(R) by type(R). From now on we abbreviate schema(R) to sch(R).

We assume a countably infinite domain of constants, *Dom*, containing a distinguished constant *unk*, denoting the null value "unknown".

A tuple over R is a total mapping t from sch(R) into *Dom* such that $\forall A_i \in$ sch(R), t(A_i) $\in$ *Dom*. The *projection* of a tuple t over R onto a set of attributes Y $\subseteq$ sch(R), denoted by t[Y], is the restriction of t to Y.

An *incomplete relation* (or simply a relation) over R is a finite set of tuples over R. A relation over R having no occurrences of *unk* is called a *complete relation.*

From now on we let R *be a relation schema and* r *is a relation over* R. As usual uppercase letters (which may be subscripted) from the end of the alphabet such as X, Y, Z will be used to denote sets of attributes, while those from the beginning of the alphabet such as A, B, D will be used to denote single attributes.

In [LEVE94, LEVE95a] we defined the semantics of incomplete relations in terms of *possible worlds* by defining a partial order, $\sqsubseteq$, in *Dom*, such that $u \sqsubseteq v$ if and only if either $u = unk$ or $u = v$, where $u, v \in Dom$. The partial order $\sqsubseteq$ is extended to tuples over R in a natural way. The set of all possible worlds relative to r, denoted by POSS(r), is the set of all complete relations that emanate from all possible substitutions of occurrences of *unk* in r by nonnull values in $Dom - \{unk\}$.

3 Functional dependencies in incomplete relations

Herein we formalise the notion a functional dependency being satisfied in an incomplete relation.

Definition 3.1 (Functional dependency) A *functional dependency* over R (or simply an FD) is a statement of the form X → Y, where X, Y ⊆ sch(R).

We call an FD of the form X → Y, where Y ⊆ X, a *trivial* FD. Two nontrivial FDs of the forms X → A and Y → A are said to be *incomparable* if X and Y are incomparable. Two nontrivial FDs of the forms XB → A and YA → B are said to be *cyclic*.

We stress the fact that we allow FDs whose left-hand side is the empty set. From now on we let F *be a set of FDs over* R. We define the *size* of an FD X → Y to be |X| + |Y|, and the *size* of F, denoted by ||F||, to be the sum of the sizes of all the FDs in F.

Definition 3.2 (Satisfaction of an FD) An FD X → Y is satisfied in a relation r, denoted by r ⊨ X → Y, whenever $\forall t_1, t_2 \in$ r, if ∀ A ∈ X, t_1[A] ≠ *unk* and t_1[X] = t_2[X] then ∀ B ∈ Y, either t_1[B] = *unk*, t_2[B] = *unk* or t_1[B] = t_2[B].

The reader can verify that when the relation, r, in Definition 3.2 is a complete relation then the definition of satisfaction of an FD in r reduces to the standard definition of satisfaction of an FD [ULLM88, MANN92, ATZE93]. It was shown in [LEVE94, LEVE95a] that X → Y is satisfied in r if and only if there exists a complete relation s ∈ POSS(r) that satisfies the FD in the standard way.

Definition 3.3 (Logical implication) A set of FDs F over R *logically implies* an FD X → Y, written F ⊨ X → Y, if whenever r is a relation over R then the following condition is true:

$$\text{if } \forall \mathrm{W} \rightarrow \mathrm{Z} \in \mathrm{F}, \mathrm{r} \models \mathrm{W} \rightarrow \mathrm{Z} \text{ holds then } \mathrm{r} \models \mathrm{X} \rightarrow \mathrm{Y} \text{ also holds.}$$

4 Some properties of sets of functional dependencies

We assume that the reader is familiar with Armstrong's axiom system for FDs [ARMS74, ULLM88, MANN92, ATZE93], consisting of the inference rules: reflexivity, augmentation and transitivity. A fundamental result in relational database theory is that Armstrong's axiom system is sound and complete for FDs holding in complete relations. We denote the closure of a set of FDs F over R with respect to Armstrong's axiom system by F^+. Lien [LIEN82], and Atzeni and Morfuni [ATZE86] have shown that the inference rules: reflexivity, augmentation, decomposition and union, are sound and complete for FDs holding in incomplete relations; we call this axiom system, Lien and Atzeni's axiom system. That is, by dropping the transitivity rule from Armstrong's axiom system and adding the decomposition and union rules, we obtain Lien and Atzeni's axiom system. We denote the closure of a set of FDs F over R with respect to Lien and Atzeni's axiom system by F^*. The soundness and completeness of Lien and Atzeni's axiom system for FDs holding in incomplete relations can be written symbolically as the statement: F $\models$ X $\rightarrow$ Y if and only if X $\rightarrow$ Y $\in F^*$.

The following useful property of derivations of FDs, using Armstrong's axiom system, which appears as Lemma 2 in [BEER79], will be used in subsequent proofs.

Proposition 4.1 Let F be a set of FDs and assume that W $\rightarrow$ Z $\in$ F is used nonredundantly in a derivation of an FD X $\rightarrow$ Y $\in F^+$ from F by using Armstrong's axiom system. Then X $\rightarrow$ W $\in$ (F $-$ {W $\rightarrow$ Z})$^+$.

Definition 4.1 (Closure of a set of attribute) The closure of a set of attributes X $\subseteq$ sch(R), with respect to a set of FDs F, denoted by C_F(X) (or simply C(X) whenever F is understood from context), is given by

$$C(\mathrm{X}) = \bigcup\{\mathrm{Y} \mid \mathrm{X} \rightarrow \mathrm{Y} \in F^+\}.$$

A set of attributes X $\subseteq$ sch(R) is *closed* with respect to F (or simply closed whenever F is understood from context) if C_F(X) = X.

We note that C(X) can be computed in linear time in the size of F [BEER79]. In the sequel we will use the equivalent statements Y $\subseteq C_F$(X) and X $\rightarrow$ Y $\in F^+$, interchangeably.

Definition 4.2 (Superkey, key and antikey) A set of attributes X $\subseteq$ sch(R) is a *superkey* for R with respect to F (or simply a superkey for R whenever F is understood from context), if C_F(X) = sch(R). A set of attributes X $\subseteq$ sch(R) is a *key* for R with respect to F (or simply a key for R whenever F is understood from context), if X is a superkey for R with respect to F and, in addition, for no proper subset Y $\subset$ X, is it the case that Y is a superkey for R with respect to F. We denote the set of all keys for R with respect to F by $\mathcal{K}$(F).

An attribute A $\in$ sch(R) is *prime* with respect to F (or simply prime whenever F is understood from context) if A $\in$ X for some X $\in \mathcal{K}$(F); otherwise A is *nonprime* with respect to F.

An *antikey* for R with respect to F (or simply an antikey for R whenever F is understood from context) is a maximal subset X of sch(R) such that X is not a superkey for R. We denote the set of all antikeys for R with respect to F by $\mathcal{A}$(F).

Definition 4.3 (A cover of a set of FDs) A set of FDs G over R is a *cover* of F if $F^+ = G^+$.

By Definition 4.1 if G is a cover of a set of FDs F then C_F(X) = C_G(X).

Definition 4.4 (Reduced and canonical sets of FDs) An FD X $\rightarrow$ Y $\in F^+$ is *reduced* [BEER79] if there does not exist a set of attributes W $\subset$ X such that W $\rightarrow$ Y $\in F^+$. A set of FDs F is *reduced* if all the FDs in F are reduced; F is *canonical* if it is reduced and the right-hand sides of all the FDs in F are singletons.

A reduced cover G of F can be obtained in polynomial time in the size of F [BEER79].

Definition 4.5 (A minimum set of FDs) A set of FDs F is a *minimum* [MAIE80] set of FDs if there is no cover G of F such that G has fewer FDs than F, all the FDs in F are reduced and for every FD X $\rightarrow$ Y $\in$ F and for every Z $\subset$ Y, ((F $-$ {X $\rightarrow$ Y}) $\cup$ {X $\rightarrow$ Z})$^+ \neq F^+$.

In [MAIE80] a minimum set of FDs is called an *LR-minimum* set of FDs. Furthermore, a minimum cover G of a set of FDs F can be obtained in polynomial time in the size of F [MAIE80].

Definition 4.6 (An optimum set of FDs) A set of FDs F is an *optimum* [MAIE80, MANN83] set of FDs if there does not exist a cover G of F such that $||G|| < ||F||$. We denote an optimum cover of a set of FDs F by opt(F).

In [MAIE80] it was shown that, in general, finding an optimum cover is NP-complete [MAIE80].

Definition 4.7 (Equivalent sets of attributes) Given a set of FDs F, the sets of attributes X, Y $\subseteq$ sch(R), are *equivalent* under F, if X $\rightarrow$ Y, Y $\rightarrow$ X $\in F^+$. We denote the subset of FDs in F whose left-hand sides are equivalent to a set of attributes X $\subseteq$ sch(R) by E_F(X); we call the sets E_F(X) the *equivalence classes* of F.

5 Monodependent Sets of Functional Dependencies

Given a set of FDs F and an incomplete relation r it is natural to say that r satisfies F if there is some complete relation, s $\in$ POSS(r), such that s satisfies each of the FDs in F. This gives rise to the *additivity problem*, which is the problem of whether the statement that r satisfies F is equivalent to the statement that r satisfies each FD in a reduced cover G of F [LEVE94, LEVE95a] (cf. [ATZE93]); if these two statements are equivalent for a class of incomplete relations and a class of sets of FDs then we say that *satisfaction is additive* with respect to these classes. If satisfaction is not additive, then F may be viewed as contradictory. Thus we consider the solution of the additivity problem to be an important prerequisite for any relational database system supporting FDs in the context of incomplete information, since otherwise semantic anomalies may arise.

Obviously satisfaction is additive with respect to the class of complete relations and the class of all sets of FDs. On the other hand, it is well known that satisfaction is *not* additive with respect to the class of incomplete relations and the class of all sets of FDs [ATZE86, LEVE94]. In [LEVE94] we introduced the class *monodependent* sets of FDs. Informally, a set of FDs F over R is monodependent if for each attribute A $\in$ sch(R), there is a unique nontrivial and reduced FD in F^+ that functionally determines A, and in addition there are no two nontrivial FDs in F^+ such that the right-hand side of each of the two FDs splits the left-hand side of the other FD. The main result in [LEVE94] shows that satisfaction is additive with respect to the class of all incomplete relations and a class of sets of FDs, **FC**, if and only if all the sets of FDs in **FC** are monodependent sets of FDs.

In [LEVE95b] we studied the impact on normalisation theory in relation databases of assuming that sets of FDs are monodependent, and in [LEVE95a] we extended the results in [LEVE94] to the wider class of sets of FDs and *unary inclusion dependencies* [COSM90].

Definition 5.1 (A monodependent set of FDs) A set of FDs F is a *monodependent* set of FDs over R (or simply monodependent whenever R is understood from context) if $\forall$ A $\in$ sch(R), the following two conditions are true:

1. Whenever there exist incomparable FDs, X $\to$ A, Y $\to$ A $\in F^+$, then X $\cap$ Y $\to$ A $\in F^+$; we call this property the *intersection property*.
2. Whenever there exist cyclic FDs, XB $\to$ A, YA $\to$ B $\in F^+$, then either Y $\to$ B $\in F^+$ or (X $\cap$ Y)A $\to$ B $\in F^+$; we call this property the *split-freeness property*.

An immediate consequence of the above definition is that if G is a cover of F then F is monodependent if and only if G is monodependent. In addition, we have shown in [LEVE94] that monodependence of a set of FDs F can be checked in polynomial time in the size of F.

We observe that the two defining properties of monodependent sets of FDs correspond to the two defining properties of *conflict-free* sets of *multivalued dependencies* (MVDs) [SCIO81, LIEN82, BEER86]. We further observe that the set of MVDs that is logically implied by a monodependent set of FDs may not be conflict-free and thus monodependence is a weaker notion than conflict-freeness. For example, let F = {A $\to$ B, B $\to$ A}, with sch(R) = {A, B, D}. It can easily be verified that R is monodependent but that the set of MVDs logically implied by R is not conflict-free.

The next theorem from [LEVE94] shows that if F satisfies the intersection property, then the closure of F with respect to Armstrong's axiom system (i.e. F^+) is equal to the closure of F with respect to Lien and Atzeni's axiom system (i.e. F^*).

This result is fundamental to the theory of FDs in incomplete relations, since it justifies the use of Armstrong's axiom system in the context of incomplete relations when F is monodependent.

Theorem 5.1 If F satisfies the intersection property then $F^+ = F^*$.

The converse of Theorem 5.1 is, in general, false. For example, let F = {A $\rightarrow$ D, B $\rightarrow$ D}, with sch(R) = {A, B, D}. It can be easily verified that $F^+ = F^*$, however, F does not satisfy the intersection property, since $\emptyset \rightarrow$ D $\notin F^+$.

The following technical results, which are utilised in the sequel, are proved in [LEVE95b].

Lemma 5.2 Let F be a set of FDs that is minimum and satisfies the intersection property. Then $\forall$ A $\in$ sch(R), there is at most one FD, X $\rightarrow$ Y $\in$ F, such that A $\in$ Y.

Lemma 5.3 Let F be a set of FDs which is monodependent and minimum, and let E_F(X) be an equivalence classes of F such that $|E_F$(X)$|>$ 1. Then any two FDs in E_F(X) are reduced and of the form, WA $\rightarrow$ B and WB $\rightarrow$ A.

Lemma 5.4 Let F be a set of FDs which is monodependent and minimum, and let E_F(X) and E_F(V) be distinct and nonempty equivalences classes of F, with W $\rightarrow$ Z $\in E_F$(X). Then $\forall$ A $\in$ WZ, A does not appear in the right-hand side of any FD in E_F(V).

Lemma 5.5 Let F be a set of FDs which is monodependent and minimum, and let $E = \{E_F(X_1), E_F(X_2), \ldots, E_F(X_k)\}$ be the set of all nonempty equivalence classes of F. Then the number of keys for R is given by

$$\mid \mathcal{K}(F) \mid = \prod_{i=1}^{k} \mid E_F(X_i) \mid .$$

Theorem 5.6 If a set of FDs F is minimum and satisfies the intersection property then it is also an optimum set of FDs.

An immediate result of Theorem 5.6 is that finding an optimum cover of a set of FDs which satisfies the intersection property can be computed in polynomial time. This is due to the fact that finding a minimum cover of a set of FDs can be computed in polynomial time [MAIE80, WILD95]. In general, when a set of FDs does not satisfy the intersection property, then finding an optimum cover is

NP-complete [MAIE80]. We note that in [LEVE95b] we have also shown that the optimal cover of a set of FDs F that satisfies the intersection property is unique, implying that the minimum cover of F is also unique.

The next corollary follows from Theorem 5.6 Lemma 5.2, and the fact that if $X \rightarrow Y$ is an FD, with $X \cap Y = \emptyset$, then the size of $X \rightarrow Y$ is less than or equal to type(R).

Corollary 5.7 If a set of FDs F is optimum and satisfies the intersection property, then $|F| \leq$ type(R) and $||F|| \leq (\text{type}(R))^2$.

We close this section with an interesting result showing that monodependent sets of FDs which are also optimum are closed under the proper subset operation.

Proposition 5.8 Let F be a monodependent set of FDs and let G = opt(F). Then $\forall$ $H \subset G$, H is a monodependent and optimum set of FDs over R.

Proof. Let $H \subset G$. By Lemmas 5.3 and 5.4, and Proposition 4.1 we can deduce that $X \rightarrow A \in G^+$ and $A \in Z$ for some FD $W \rightarrow Z \in H$ if and only if $X \rightarrow A \in H^+$. We call this statement Observation 1. Therefore, H must satisfy the intersection property, since otherwise there must exist incomparable FDs $X \rightarrow A$, $Y \rightarrow A \in H^+$, but $X \cap Y \rightarrow A \in G^+ - H^+$, which contradicts Observation 1. Similarly, H must satisfy the split-freeness property, since otherwise there must exist cyclic FDs, $XB \rightarrow A$, $YA \rightarrow B \in H^+$, but either $Y \rightarrow B \in G^+ - H^+$ or $(X \cap Y)A \rightarrow B \in G^+ - H^+$, which again contradicts Observation 1.

Next, suppose that H is not optimum and that J = opt(H), with $||J||<||H||$. Therefore, $((G - H) \cup J)^+ = G^+$. This leads to a contradiction that G is optimum, since $||(G - H) \cup J||<||G||$. The result that H is optimum follows. □

6 The Lattice of Closed Sets

Herein we give the definitions of the lattice-theoretic concepts used in the rest of the paper. The reader is referred to [DAVE90] for an introduction to lattice theory and to [GRAT78] for more advanced material.

The operator C_F (see Definition 4.1) which closes sets of attributes in sch(R) is a *closure operator* in the lattice-theoretic sense [DAVE90]. It follows by [DAVE90, Theorem 2.21] that the family of all the closed sets in the power set of sch(R) is a lattice partially ordered by set inclusion, which we denote by $\mathcal{L}(F)$ (see also [DEME92, DEME93]). The lattice $\mathcal{L}(F)$ is, by definition, cover insensitive and thus G is a cover of F if and only if $\mathcal{L}(F) = \mathcal{L}(G)$. It is easy to see that $\mathcal{L}(F)$ is closed under intersection and thus the greatest lower bound of two closed sets in $\mathcal{L}(F)$ is just their intersection. On the other hand, it can be verified that the the least upper bound, denoted by $\sqcup$, of two closed sets $X, Y \in \mathcal{L}(F)$ is given by $X \sqcup Y = C(X \cup Y)$. We refer the reader to [DEME92] for an in-depth investigation concerning the connection between the structure of a set of FDs and the type of lattice of closed sets it induces.

The following result shown in [DEME92, DEME93] shows the basic connection between a set of FDs F over R and its induced lattice of closed sets $\mathcal{L}(\mathrm{F})$.

Proposition 6.1 There is a one-to-one correspondence between F^+ and $\mathcal{L}(\mathrm{F})$.

Definition 6.1 (Meet-irreducible elements) A closed set X $\in \mathcal{L}(\mathrm{F})$ is *meet-irreducible* [DAVE90] if $\forall$ Y, Z $\in \mathcal{L}(\mathrm{F})$, X = Y $\cap$ Z implies that either X = Y or X = Z. The family of all meet-irreducible closed sets in $\mathcal{L}(\mathrm{F})$ is denoted by $\mathcal{M}(\mathrm{F})$.

The following result shows the basic connection between $\mathcal{L}(\mathrm{F})$ and $\mathcal{M}(\mathrm{F})$ [BEER84, MANN86, WILD95].

Proposition 6.2 $\mathcal{M}(\mathrm{F})$ is the unique minimal subset of $\mathcal{L}(\mathrm{F})$ such that X $\in \mathcal{L}(\mathrm{F})$ if and only if X is the intersection of all the closed sets in $\mathcal{M}(\mathrm{F})$ that are supersets of X.

The following result, which was shown in [MANN86], gives an alternative characterisation of $\mathcal{M}(\mathrm{F})$.

Lemma 6.3 Let MAX(F, A) be the family of all the maximal closed sets $\mathcal{L}(\mathrm{F})$ such that $\forall$ X $\in$ MAX(F, A), A $\notin$ X. Then the following equality holds:

$$\mathcal{M}(\mathrm{F}) = \bigcup_{\mathrm{A} \in \mathrm{sch(R)}} \mathrm{MAX(F, A)}.$$

For completeness of the paper we include the definitions of the various types of lattices referred to hereafter. In particular, we define distributive, modular [GRAT78, DAVE90], semimodular [GRAT78], exchange [GRAT78] and antiexchange [JAMI85] lattices.

Definition 6.2 (Distributive lattice) $\mathcal{L}(\mathrm{F})$ is *distributive* if

$$\forall X, Y, Z \in \mathcal{L}(\mathrm{F}), X \cap (Y \sqcup Z) = (X \cap Y) \sqcup (X \cap Z).$$

Definition 6.3 (Semimodular and modular lattice) We say that X is *covered by* Y, denoted by X $\prec$ Y, where X, Y $\in \mathcal{L}(\mathrm{F})$, if X $\subset$ Y and X $\subseteq$ Z $\subset$ Y implies that Z = X, with Z $\in \mathcal{L}(\mathrm{F})$.

$\mathcal{L}(\mathrm{F})$ satisfies the *upper covering condition* if

$$\forall X, Y, Z \in \mathcal{L}(\mathrm{F}), X \prec Y \text{ implies that } X \sqcup Z \prec Y \sqcup Z \text{ or } X \sqcup Z = Y \sqcup Z.$$

The *lower covering condition* is the dual statement of the upper covering condition.

$\mathcal{L}(\mathrm{F})$ is *semimodular* if it satisfies the upper covering condition. $\mathcal{L}(\mathrm{F})$ is *modular* if it satisfies both the upper and lower covering conditions.

Definition 6.4 (Exchange property) $\mathcal{L}(\mathrm{F})$ satisfies the *exchange property* (or simply $\mathcal{L}(\mathrm{F})$ is exchange) whenever

$$\forall A, B \in \mathrm{sch}(\mathrm{R}), \forall X \subseteq \mathrm{sch}(\mathrm{R}), \text{ if } A, B \notin C(X) \text{ and } A \in C(XB) \text{ then } B \in C(XA).$$

Definition 6.5 (Antiexchange property) $\mathcal{L}(\mathrm{F})$ satisfies the *antiexchange property* (or simply $\mathcal{L}(\mathrm{F})$ is antiexchange) whenever

$$\forall A, B \in \mathrm{sch}(\mathrm{R}), \forall X \subseteq \mathrm{sch}(\mathrm{R}), \text{ if } A, B \notin C(X) \text{ and } A \in C(XB) \text{ then } B \notin C(XA).$$

The reader can also verify that the intersection property can be redefined as follows in terms of a property of the lattice $\mathcal{L}(\mathrm{F})$ of closed sets.

Definition 6.6 (Intersection property) Let $\div$ be the symmetric difference operator, i.e. $\mathrm{X} \div \mathrm{Y} = (\mathrm{X} - \mathrm{Y}) \cup (\mathrm{Y} - \mathrm{X})$, where X, Y $\subseteq$ sch(R). Then $\mathcal{L}(\mathrm{F})$ satisfies the *intersection property* if

$$C(X \cap Y) - (X \div Y) = (C(X) \cap C(Y)) - (X \div Y).$$

The reader can also verify that the split-freeness property can be redefined as follows in terms of a property of the lattice $\mathcal{L}(\mathrm{F})$ of closed sets.

Definition 6.7 (Split-freeness property) $\mathcal{L}(\mathrm{F})$ satisfies the *split-freeness property* if

$$\forall A, B \in \mathrm{sch}(\mathrm{R}), \forall X, Y \subseteq \mathrm{sch}(\mathrm{R}), \text{ if } B \in C(YA), B \notin C(Y)$$

$$\text{and } B \notin C((X \cap Y)A) \text{ then } A \notin C(XB).$$

We say that a lattice $\mathcal{L}(\mathrm{F})$ embeds the figure $\mathcal{N}$, if $\exists$ W, X, Y, Z $\in \mathcal{L}(\mathrm{F})$ such that W $\subset$ X, Y $\subset$ X and Y $\subset$ Z. It can be verified that if $\mathcal{L}(\mathrm{F})$ does not embed the figure $\mathcal{N}$, then $\mathcal{L}(\mathrm{F})$ satisfies the split-freeness property.

When F is a monodependent set of FDs, we say that $\mathcal{L}(\mathrm{F})$ is *monodependent*. We close this section by defining the concept of a sublattice.

Definition 6.8 (Sublattice) A subset S $\subseteq \mathcal{L}(\mathrm{F})$ is a *sublattice* [DAVE90] of $\mathcal{L}(\mathrm{F})$ if X, Y $\in$ S implies that both X $\sqcup$ Y $\in$ S and X $\sqcap$ Y $\in$ S.

7 Counterexamples for monodependent set of FDs

Herein we present some negative results concerning the structure of $\mathcal{L}$(F) when F is monodependent. We first show that $\mathcal{L}$(F) may not even be semimodular and that a distributive lattice of closed sets may not be monodependent. We then show that, in general, the concepts of monodependence and exchange, and also of monodependence and antiexchange are incomparable.

Proposition 7.1 The lattice of closed sets of a monodependent set of FDs F is *not*, in general, semimodular.

Proof. Let R be a relation schema with sch(R) = {A, B, D} and let F = {AB $\rightarrow$ D, BD $\rightarrow$ A}. It can easily be verified that F is monodependent. Furthermore, $\emptyset \prec$ B but it is *not* true that ($\emptyset \sqcup$ A = A) $\prec$ (A $\sqcup$ B = ABD). Therefore, $\mathcal{L}$(F) is not semimodular. □

In [WILD89] it was shown that when $\mathcal{L}$(F) is modular then an optimum cover of F can be obtained in polynomial time in the size of F. By Proposition 7.1, Theorem 5.6 is incomparable with Wild's result, since $\mathcal{L}$(F) may satisfy the intersection property but not be modular. Furthermore as the next example shows $\mathcal{L}$(F) being modular does not imply that a minimum cover of F is also optimum.

Example 7.1 Let F = {D $\rightarrow$ AB, E $\rightarrow$ AB, AB $\rightarrow$ DE}, with sch(R) = {A, B, D, E}. It can easily be verified that $\mathcal{L}$(F) is modular and that F is minimum. On the other hand, F is not optimum, since it can be verified that G = {D $\rightarrow$ AB, E $\rightarrow$ D, AB $\rightarrow$ DE} is an optimum cover of F.

Proposition 7.2 Distributive lattices of closed sets are *not*, in general, monodependent.

Proof. We give two counterexamples of a relation schema R and a set of FDs F such that $\mathcal{L}$(F) is distributive but F is not monodependent. In the first example F violates the intersection property and in the second example F violates the split-freeness property.

Counterexample 1. Let R be a relation schema with sch(R) = {A, B, D} and let F = {B $\rightarrow$ A, D $\rightarrow$ A}. It can easily be verified that $\mathcal{L}$(F) is distributive. Furthermore, the set of FDs F is not monodependent, since it violates the intersection property due to the fact that $\emptyset \rightarrow$ A $\notin F^+$.

Counterexample 2. Let R be a relation schema with sch(R) = {A, B, D} and let F = {B $\rightarrow$ AD, AD $\rightarrow$ B}. It can easily be verified that $\mathcal{L}$(F) is distributive. Furthermore, the set of FDs F is not monodependent, since it violates the split-freeness property due to the fact that both A $\rightarrow$ B $\notin F^+$ and D $\rightarrow$ B $\notin F^+$. □

Proposition 7.3 The lattice of closed sets of a monodependent set of FDs F is *not*, in general, exchange.

Proof. Let R be a relation schema with sch(R) = {A, B} and let F = {A $\rightarrow$ B}. It can easily be verified that F is monodependent. Furthermore, A,B $\notin C(\emptyset)$ and A $\in C$(B) but B $\notin C$(A). $\square$

Proposition 7.4 Exchange lattices of closed sets are *not*, in general, monodependent.

Proof. Let R be a relation schema with sch(R) = {A, B, D} and let F = {A $\rightarrow$ B, B $\rightarrow$ A, A $\rightarrow$ D, D $\rightarrow$ A}. It can easily be verified that $\mathcal{L}$(F) is exchange. Furthermore, the set of FDs F is not monodependent, since it violates the intersection property. $\square$

Proposition 7.5 The lattice of closed sets of a monodependent set of FDs F is *not*, in general, antiexchange.

Proof. Let R be a relation schema with sch(R) = {A, B} and let F = {A $\rightarrow$ B, B $\rightarrow$ A}. It can easily be verified that F is monodependent. Furthermore, A,B $\notin C(\emptyset)$, A $\in C$(B) and also B $\in C$(A). $\square$

Proposition 7.6 Antiexchange lattices of closed sets are *not*, in general, monodependent.

Proof. Let R be a relation schema with sch(R) = {A, B, D} and let F = {B $\rightarrow$ A, D $\rightarrow$ A}. It can easily be verified that $\mathcal{L}$(F) is antiexchange. Furthermore, the set of FDs F is not monodependent, since it violates the intersection property. $\square$

8 The connection between exchange and antiexchange lattices and monodependence

Herein we investigate the connection between exchange and antiexchange lattices of closed sets, and sets of FDs that satisfy the split-freeness property. We first show that if F satisfies the intersection property and $\mathcal{L}$(F) is either exchange or antiexchange then F is monodependent. We then show that when F satisfies the intersection property then $\mathcal{L}$(F) is exchange if and only if F satisfies the split-freeness property and the cardinality of all the nonempty equivalence classes of F is maximal, i.e. for each such equivalence class the said cardinality is the size of any FD in the class (see Lemma 5.3). Finally we show that when F satisfies the intersection property then $\mathcal{L}$(F) is antiexchange if and only if F satisfies the split-freeness property and the cardinality of all the nonempty equivalence classes of F is minimal, i.e. the said cardinality is one. We conclude that the structure of the lattice of closed sets of a monodependent set of FDs is something in between an exchange and antiexchange lattice according to the cardinalities of its equivalence classes.

Several properties of exchange and antiexchange lattices of closed sets have been investigated in [DEME92]. When $\mathcal{L}$(F) is exchange then *Boyce-Codd normal form* [ULLM88, MANN92, ATZE93] can be characterised in terms of a *uniform closure.* In addition, if $\mathcal{L}$(F) is exchange and $C(\emptyset) = \emptyset$, then second normal form and third normal form are equivalent. (See [ULLM88, MANN92, ATZE93] for the definitions of the various normal forms.) When $\mathcal{L}$(F) is antiexchange then for every subset X $\subseteq$ sch(R), there is a unique reduced FD such that Y $\rightarrow$ X $\in F^+$. In particular, when $\mathcal{L}$(F) is antiexchange, then $|\mathcal{K}$(F)$|= 1$ [BISK91].

Lemma 8.1 Let F be a set of FDs that satisfies the intersection property. Then if $\mathcal{L}$(F) is either exchange or antiexchange, then F satisfies the split-freeness property, i.e. F is monodependent.

Proof. Assume to the contrary that F does not satisfy the split-freeness property. Therefore, by Definition 5.1 there exist cyclic FDs, XB $\rightarrow$ A, YA $\rightarrow$ B $\in F^+$, but both Y $\rightarrow$ B $\notin F^+$ and (X $\cap$ Y)A $\rightarrow$ B $\notin F^+$. We can assume without loss of generality that XB $\rightarrow$ A and YA $\rightarrow$ B are reduced FDs. Thus it is also the case that X $\rightarrow$ A $\notin F^+$. Now, Y $\not\subseteq$ X holds, otherwise (X $\cap$ Y)A $\rightarrow$ B is simply YA $\rightarrow$ B, which is assumed to be in F^+. There are two case to consider.

Firstly, assume that X $\subset$ Y and thus YB $\rightarrow$ A $\in F^+$ but XA $\rightarrow$ B $\notin F^+$, since the FDs are reduced. Thus $\mathcal{L}$(F) is not exchange, since A, B $\notin C$(X) and A $\in C$(XB) but B $\notin C$(XA). Furthermore, $\mathcal{L}$(F) is not antiexchange, since A, B $\notin C$(Y) and B $\in C$(YA) but also A $\in C$(YB).

Secondly, assume that X and Y are incomparable. Now, we have that A, B $\notin C$(X) and A $\in C$(XB). Assume that $\mathcal{L}$(F) is exchange and thus B $\in C$(XA). Thus, XA $\rightarrow$ B $\in F^+$ and YA $\rightarrow$ B $\in F^+$ are incomparable FDs. It follows that (X $\cap$ Y)A $\rightarrow$ B $\in F^+$ by the intersection property, which contradicts the fact that F does not satisfy the split-freeness property. Thus B $\notin C$(XA) and $\mathcal{L}$(F) is not exchange. Now, if A $\in C$(XY), then X $\rightarrow$ A $\in F^+$ by the intersection property, and similarly if B $\in C$(XY), then Y $\rightarrow$ B $\in F^+$ also by the intersection property. If X $\rightarrow$ A $\in F^+$ then fact that XB $\rightarrow$ A is reduced is contradicted, and correspondingly, if Y $\rightarrow$ B $\in F^+$ then the fact that YA $\rightarrow$ B is reduced is contradicted. So, we conclude that A, B $\notin C$(XY). It follows that $\mathcal{L}$(F) is not antiexchange, since A $\in C$(XYB) but also B $\in C$(XYA). The result that F satisfies the split-freeness property and is thus monodependent follows as required. □

Theorem 8.2 Let F be a set of FDs that satisfies the intersection property, and let $E = \{E_G(X_1), E_G(X_2), \ldots, E_G(X_k)\}$ be the set of all nonempty equivalence classes of G, where $G = \text{opt}(F)$. Then the following statements are equivalent:

1. $\mathcal{L}$(F) is exchange.

2. F satisfies the split-freeness property, i.e. F is monodependent, and $\forall E_G(X_i) \in E, | E_G(X_i) | = | XY |$, for some FD X $\rightarrow$ Y $\in E_G(X_i)$.

Proof. (1 $\Rightarrow$ 2.) By Lemma 8.1 F satisfies the split-freeness property. Now, assume that for some $E_G(X_i) \in$ E and X $\rightarrow$ Y $\in E_G(X_i)$, $|E_G(X_i)|<$|XY|. Thus, by Lemma 5.3, $\exists$ A, B $\in$ sch(R) such that WB $\rightarrow$ A $\in E_G(X_i)$ but WA $\rightarrow$ B $\notin E_G(X_i)$. Furthermore, also by Lemma 5.3, W $\rightarrow$ A, W $\rightarrow$ B $\notin F^+$, leading to a contradiction of the fact that $\mathcal{L}$(F) is exchange.

(2 $\Rightarrow$ 1.) Suppose to the contrary that $\mathcal{L}$(F) is not exchange. Then for some set of attributes, V $\subset$ sch(R), $\exists$ A, B $\in$ sch(R) such that A, B $\notin C$(V), A $\in C$(VB) but B $\notin C$(VA). Thus VB $\rightarrow$ A $\in F^+$ and there is some equivalence class E_G(X) $\in$ E such that X $\rightarrow$ Y $\in E_G$(X), with A $\in$ Y. If |XY| = 1, then X = $\emptyset$ and V $\rightarrow$ A $\in F^+$, leading to a contradiction. So, it must be the case that |XY|$>$ 1 and thus by Lemma 5.3 Y = {A}. Now, if B $\notin$ X, then by the intersection property, it follows that (X $\cap$ VB) $\rightarrow$ A $\in F^+$ and thus V $\rightarrow$ A $\in F^+$, since B $\notin$ (X $\cap$ VB). So, it must be the case that B $\in$ X and thus (X$-$B)A $\rightarrow$ B $\in E_G$(X), since $|E_G$(X)|=|XA|. Let W = (X$-$B) $\cap$ V. Then by the intersection property WB $\rightarrow$ A $\in F^+$, with W $\subseteq$ V. Furthermore, by Lemma 5.3, X = WB, since X $\rightarrow$ A is reduced. Thus WA $\rightarrow$ B $\in E_G$(X) and VA $\rightarrow$ B $\in F^+$ leading to a contradiction. It follows that B $\in C$(VA) and thus $\mathcal{L}$(F) is exchange as required. □

Theorem 8.3 Let F be a set of FDs that satisfies the intersection property, and let $E = \{E_G(X_1), E_G(X_2), \ldots, E_G(X_k)\}$ be the set of all nonempty equivalence classes of G, where G = opt(F). Then the following statements are equivalent:

1. $\mathcal{L}$(F) is antiexchange.

2. F satisfies the split-freeness property, i.e. F is monodependent, and $\forall E_G(X_i) \in$ E, $|E_G(X_i)|=1$.

Proof. (1 $\Rightarrow$ 2.) By Lemma 8.1 F satisfies the split-freeness property. Furthermore, by [JAMI85, DEME92] $|\mathcal{K}$(F)|= 1, since $\mathcal{L}$(F) is antiexchange. Now, assume that for some $E_G(X_i) \in$ E, $|E_G(X_i)|>1$. Thus, by Lemma 5.5 $|\mathcal{K}$(F)|$>$ 1, leading to a contradiction of the fact that $\mathcal{L}$(F) is antiexchange.

(2 $\Rightarrow$ 1.) Suppose to the contrary that $\mathcal{L}$(F) is not antiexchange. Then for some set of attributes, V $\subset$ sch(R), $\exists$ A, B $\in$ sch(R) such that A, B $\notin C$(V), A $\in C$(VB) but also B $\in C$(VA). Thus VA $\rightarrow$ B, VB $\rightarrow$ A $\in F^+$. Now, there exist equivalence classes $E_G(X_1), E_G(X_2) \in$ E such that X $\rightarrow$ Y $\in E_G(X_1)$, with A $\in$ Y, and W $\rightarrow$ Z $\in E_G(X_2)$, with B $\in$ Z.

Assume that $E_G(X_1) = E_G(X_2)$. There are two cases to consider. Firstly, if X $\rightarrow$ Y and W $\rightarrow$ Z are distinct FDs then $|E_G(X_1)|> 1$, leading to a contradiction. Secondly, if X $\rightarrow$ Y and W $\rightarrow$ Z are in fact the same FD, then X $\rightarrow$ AB $\in F^+$, with A, B $\notin$ X. Now, VA $\nsubseteq$ X, since A $\notin$ X, and also X $\nsubseteq$ VA, since otherwise X $\subseteq$ V and V $\rightarrow$ A $\in F^+$ leading to a contradiction. Therefore, X $\rightarrow$ B $\in F^+$ and VA $\rightarrow$ B $\in F^+$ are incomparable FDs. It follows by the intersection property that (X

$\cap$ VA) $\rightarrow$ B $\in F^+$, with (X $\cap$ VA) $\subseteq$ V. Therefore, V $\rightarrow$ B $\in F^+$ again leading to a contradiction.

So, assume that $E_G(X_1)$ and $E_G(X_2)$ are distinct equivalence classes of G. Thus, by Lemma 5.4, A $\notin$ WZ and B $\notin$ XY. Now, VA $\not\subseteq$ W, since A $\notin$ W, and also W $\not\subseteq$ VA, since otherwise W $\subseteq$ V and V $\rightarrow$ B $\in F^+$ leading to a contradiction. Therefore, W $\rightarrow$ B $\in F^+$ and VA $\rightarrow$ B $\in F^+$ are incomparable FDs. It follows by the intersection property that (W $\cap$ VA) $\rightarrow$ B $\in F^+$, with (W $\cap$ VA) $\subseteq$ V. Therefore, V $\rightarrow$ B $\in F^+$ again leading to a contradiction. It thus follows that B $\notin C$(VA) and thus $\mathcal{L}$(F) is antiexchange as required. □

9 Characteristics of lattices satisfying the intersection property

Herein we investigate some of the characteristics of lattices of closed sets of FDs that satisfy the intersection property. We first utilise the concept of an interval, which is defined below, to investigate how the lattice of closed sets changes from one that does not necessarily satisfy the intersection property to one that does (cf. [BURO87]). We then give a characterisation of the intersection property in terms of the existence of certain distributive sublattices of $\mathcal{L}$(F). We also present a polynomial time algorithm in the size of F in order to compute the set of meet-irreducible closed sets, $\mathcal{M}$(F), when F satisfies the intersection property. As a corollary of this algorithm we show that when F satisfies the intersection property then the cardinality of $\mathcal{M}$(F) is at most $\binom{\text{type(R)}}{\text{type(R)}-2}$. As an additional corollary of this algorithm we show that testing whether an attribute is prime can be done in polynomial time in the size of F, when F satisfies the intersection property; in general, the problem of testing whether an attribute is prime is known to be NP-complete [LUCC78].

Definition 9.1 (Intersection property descriptor) The *intersection property descriptor* of a set of FDs F over a relation schema R, denoted by $\mathcal{I}$(F), is defined by

$$\mathcal{I}(F) = \{X \cap Y \rightarrow A \mid \text{ there exist incomparable FDs,}$$
$$X \rightarrow A, Y \rightarrow A \in F^+, \text{ but } X \cap Y \rightarrow A \notin F^+\}.$$

The next lemma, which characterises the lattice of closed sets of a set of FDs that satisfies the intersection property, follows from Definition 6.6 and [DEME92, Theorem 3.1]. We begin by defining the concept of an interval.

Definition 9.2 (Interval) The *interval* between X and Y, where X $\subseteq$ Y $\subseteq$ sch(R), denoted by [X, Y], is given by [X, Y] = {Z | X $\subseteq$ Z $\subseteq$ Y}.

Lemma 9.1 Given a set of FDs F, let G = F $\cup$ $\mathcal{I}$(F). Then the lattice $\mathcal{L}$(G) of closed sets of G is given by

$$\begin{aligned}\mathcal{L}(G) &= \mathcal{L}(F) - \bigcup_{X\cap Y\to A\in\mathcal{I}(F)} [X\cap Y, \mathrm{sch}(R) - A] \\ &= \mathcal{L}(F) - \bigcup_{X\cap Y\to A\in\mathcal{I}(F)} \bigcup_{\substack{B\in(X-Y)\\ D\in(Y-X)}} [X\cap Y, \mathrm{sch}(R) - ABD].\end{aligned}$$

A closed set X $\in$ S, where S $\subseteq$ $\mathcal{L}$(F), is *maximum* if $\forall$ Y $\in$ S, Y $\subseteq$ X.

Lemma 9.2 Let F be a set of FDs and $\mathcal{H}$ be the family of closed sets defined by

$$\mathcal{H} = \mathcal{L}(F) \cap \bigcup_{X\cap Y\to A\in\mathcal{I}(F)} \bigcup_{\substack{B\in(X-Y)\\ D\in(Y-X)}} [X\cap Y, \mathrm{sch}(R) - ABD].$$

Then all the maximum elements in $\mathcal{H}$ are in $\mathcal{M}$(F), i.e. all the maximum elements in $\mathcal{H}$ are meet-irreducible closed sets in $\mathcal{L}$(F).

Proof. Let X be a maximum element of $\mathcal{H}$ and Y, Z $\in$ $\mathcal{L}$(F) be two closed sets such that X = Y $\cap$ Z. We need to show that either X = Y or X = Z. Suppose to the contrary that X $\neq$ Y and X $\neq$ Z and thus both X $\subset$ Y and X $\subset$ Z hold. Now, by Lemma 9.1, Y, Z $\in$ $\mathcal{L}$(G), where G = F $\cup$ $\mathcal{I}$(F). A contradiction has arisen, since it must be the case that X $\in$ $\mathcal{L}$(G), due to the fact that $\mathcal{L}$(G) is closed under intersection. □

Definition 9.3 (The family of left-hand sides of a set of FDs) The *family of left-hand sides* of a set of FDs F with respect to A $\in$ sch(R), denoted by F(A), is defined by

$$F(\mathrm{A}) = \{\mathrm{X} \mid \mathrm{X} \to \mathrm{A} \in F^+ \text{ is a nontrivial FD}\}.$$

The schema of F(A), denoted by sch(F(A)), is defined by

$$\mathrm{sch}(F(\mathrm{A})) = \bigcup\{\mathrm{X} \mid \mathrm{X} \in F(\mathrm{A})\}.$$

We observe that F(A) $\subseteq$ $\mathcal{P}$(sch(R)) $-$ $\mathcal{L}$(F). In other words, the family of left-hand sides of F with respect to A is a subset of the complement of the lattice of closed sets of F.

Definition 9.4 (Lattice of sets) A *lattice of sets* over a finite set S is a subset of the power set, $\mathcal{P}$(S), which is closed under union and intersection [DAVE90].

The *if part* of the next theorem follows from the definition of the intersection property and the *only if part* of the theorem follows from Definition 9.3.

Theorem 9.3 A set of FDs F satisfies the intersection property if and only if $\forall$ A $\in$ sch(R), F(A) is a lattice of sets over sch(F(A)).

An immediate consequence of Theorem 9.3 is that F(A) is distributive, since it is well known that a lattice of sets over S is distributive [DAVE90].

We now give the pseudo-code of an algorithm, designated MEET_IRR(F), which will be shown to return the family $\mathcal{M}$(F) of meet-irreducible closed sets of $\mathcal{L}$(F), where F satisfies the intersection property.

Algorithm 1 (MEET_IRR(F))

```
1.  begin
2.    Meet_irr := ∅;
3.    G := opt(F);
4.    for each A ∈ sch(R) do
5.      if ∄ X → Y ∈ G, with A ∈ Y then
6.        Meet_irr := Meet_irr ∪ {sch(R)−A};
7.      else
8.        let X → Y be the FD in G with A ∈ Y;
9.        for each B ∈ X do
10.         Meet_irr := Meet_irr ∪ (sch(R)−AB);
11.       end for
12.     end if
13.   end for
14.   return Meet_irr;
15. end.
```

On using Corollary 5.7 the reader can verify that Algorithm 1 executes in polynomial time in type(R). The next theorem establishes the correctness of Algorithm 1.

Theorem 9.4 If F is a set of FDs that satisfies the intersection property, then Algorithm 1 returns $\mathcal{M}$(F).

Proof. We need to show that M = $\mathcal{M}$(F), where M = MEET_IRR(F) is the set returned by Algorithm 1.

$M \subseteq \mathcal{M}(F)$. Let W $\in$ M. By Lemma 6.3 it remains to show that W $\in$ MAX(F, A) for some A $\in$ sch(R). Consider the *for loop* beginning at line 4 and ending at line 13, with A $\in$ sch(R). If W was added to M at line 6, then the condition of the *if statment* beginning at line 5 is true, and obviously W = sch(R)−A $\in$ MAX(F, A). Otherwise, let W = sch(R)−AB be the set added to M at line 10. Now, W $\rightarrow$ A $\notin F^+$, otherwise by the intersection property W $\cap$ X $\rightarrow$ A $\in F^+$, with |W $\cap$ X| < |X|, contradicting the fact that G is optimum. Furthermore, W is a maximal set of attributes such that W $\rightarrow$ A $\notin F^+$, since X $\subseteq$ WB.

$\mathcal{M}(F) \subseteq M$. Let W $\in \mathcal{M}$(F). Then by Lemma 6.3 W $\in$ MAX(F, A) for some A $\in$ sch(R). It remains to show that W $\in M$. Suppose to the contrary that W $\notin M$.

Consider the *if statement* beginning at line 5 and ending at line 12. There are two cases to consider. Firstly the condition of line 5 is true and thus $\nexists$ X $\rightarrow$ Y $\in$ G, with A $\in$ Y. It follows that W $\subset$ sch(R)$-$A and thus it is not a maximal subset of sch(R) such that W $\rightarrow$ A $\notin F^+$, contradicting the fact that W $\in$ MAX(F, A). Secondly the condition of line 5 is false and thus by Lemma 5.2 there is a unique X $\rightarrow$ Y $\in$ F, with A $\in$ Y. Let X $\rightarrow$ Y be the FD, with A $\in$ Y, that is chosen in line 8. Therefore, by the *for loop* beginning at line 9 and ending at line 11, it follows that either X $\subset$ W or W $\subset$ Z, for some Z $\in M$, due to the fact that $|$W$|\leq$ type(R)$-$2 and A $\notin$ W. Both cases lead to a contradiction of the fact that W $\in$ MAX(F, A). The result that $M = \mathcal{M}$(F) follows. □

The next corollary, which gives a polynomial upper bound in type(R) for the cardinality of $\mathcal{M}$(F), is an immediate consequence of Theorem 9.4 on inspecting Algorithm 1. In general, when a set of FDs does not satisfy the intersection property, the cardinality of $\mathcal{M}$(F) may be exponential in type(R) [BEER84, MANN86].

Corollary 9.5 If a set of FDs F satisfies the intersection property, then $|\mathcal{M}(F)| \leq \binom{\mathrm{type(R)}}{\mathrm{type(R)}-2}$.

An immediate consequence of Corollary 9.5 is that an Armstrong relation having $\binom{\mathrm{type(R)}}{\mathrm{type(R)}-2} + 1$ tuples can be generated [MANN86]. The following result, which is immediate from Theorem 9.4 and Proposition 5.8, shows that when removing an FD X $\rightarrow$ Y from an optimum set of FDs that satisfies the intersection property, then the sets MAX(F, A) attain their simplest structure.

Corollary 9.6 Let F be set of FDs that is optimum and satisfies the intersection property. Then $\forall$ X $\rightarrow$ Y $\in$ F, $\forall$ A $\in$ Y, MAX(F $-$ {X $\rightarrow$ Y}, A) = {sch(R)$-$A}.

In general, the problem of deciding whether an attribute A $\in$ sch(R) is prime with respect to F is known to be NP-complete [LUCC78]. Our next result shows that when F satisfies the intersection property this problem, known as the prime attribute problem, can be decided in polynomial time in the size of F.

Corollary 9.7 If a set of FDs F satisfies the intersection property, then deciding whether an attribute A $\in$ sch(R) is prime can be solved in polynomial time in the size of F.

Proof. By [MANN89, Theorem 2] an attribute A $\in$ sch(R) is prime with respect to F if and only if for some W $\in$ MAX(F, A), C(WA) = sch(R); recall that C(WA) can be computed in linear time in the size of F [BEER79]. Furthermore, by [MANN89, Lemma 1], given a set of attributes X $\subseteq$ sch(R), testing whether X $\in$ MAX(F, A) can be done in polynomial time in the size of F. The result now follows by Corollary 9.5 on using Algorithm 1 to compute $\mathcal{M}$(F). □

The next theorem gives a characterisation of the intersection property in terms of the cardinality of the elements in $\mathcal{M}$(F).

Theorem 9.8 A set of FDs F satisfies the intersection property if and only if $\forall$ X $\in \mathcal{M}$(F), $|X| \geq$ type(R)-2.

Proof. The *only if* part of the theorem is an immediate consequence of Theorem 9.4 on inspecting Algorithm 1.

We prove the *if* part of the theorem by contraposition. Suppose that F violates the intersection property. Therefore, for some A $\in$ sch(R), there exist incomparable FDs, X $\rightarrow$ A, Y $\rightarrow$ A $\in F^+$, but X $\cap$ Y $\rightarrow$ A $\notin F^+$. We can assume without loss of generality that X $\rightarrow$ A and Y $\rightarrow$ A are reduced FDs.

Now, since X and Y are incomparable there is an attribute B $\in$ X$-$Y and an attribute D $\in$ Y$-$X. Let W = sch(R)$-$ABD, and thus $|W|$ = type(R)-3. There are two cases to consider.

Firstly, W $\rightarrow$ A $\in F^+$ and thus there exists some Z $\subset$ W such that Z $\in$ MAX(F, A); note that $\emptyset \rightarrow$ A $\notin F^+$, since X $\cap$ Y $\rightarrow$ A $\notin F^+$. The result now follows by Lemma 6.3.

Secondly, W $\rightarrow$ A $\notin F^+$ and thus by the construction of W we have that W $\in$ MAX(F, A). The result now follows by Lemma 6.3. □

The next proposition establishes which meet-irreducible elements of $\mathcal{L}$(F) are antikeys (in [DEME92] antikeys are called *coatoms*).

Proposition 9.9 A set of attributes X $\subset$ sch(R) is an antikey for R if and only if X is a maximal set in $\mathcal{M}$(F).

It follows from Theorem 9.4 on using Proposition 9.9 that the set of antikeys for R, i.e. $\mathcal{A}$(F), can be computed in polynomial time in type(R); in general, computing $\mathcal{A}$(F) can only be done in exponential time in type(R) [THI86]. We observe that an alternative proof to Corollary 9.7 can utilise a result in [DEME87] which states that an attribute A $\in$ sch(R) is nonprime if and only if A is a member of the intersection of all the antikeys in $\mathcal{A}$(F).

The next result establishes the connection between superkeys and antikeys [DEME88].

Proposition 9.10 A set of attributes X $\subset$ sch(R) is a superkey for R if and only if $\forall$ Y $\in \mathcal{A}$(F), X $\not\subseteq$ Y.

In [DEME88] Proposition 9.10 is used to derive an algorithm which computes the set of all keys $\mathcal{K}$(F) given the set of all antikeys $\mathcal{A}$(F).

In general, the problem of finding a superkey for R with respect to F, whose cardinality is less than or equal to a natural number k, is known to be NP-complete [LUCC78, DEME88]. Our next result shows that this problem, known as the superkey of cardinality k problem, is still NP-complete when F satisfies the intersection property

Theorem 9.11 The superkey of cardinality k problem is NP-complete, when F is a set of FDs that satisfies the intersection property.

Proof. The problem is known to be in NP [LUCC78]. It remains to show that the problem is NP-hard.

By [DEME88, Lemma 2.4] the vertex cover problem, which is known to be NP-complete [GARE79], can be reduced to the following problem. Given a set of antikeys for R, say **S**, such that $\forall$ X $\in$ **S**, |X| = type(R)−2, solve the superkey of cardinality k problem.

By the remark made after Proposition 9.10 it follows that **S** can be used to derive the set of keys $\mathcal{K}$(F) for some set of FDs F over R. Furthermore, we can assume, without loss of generality, that the set {X $\rightarrow$ sch(R) | X $\in \mathcal{K}$(F)} is a cover of F. It follows by Proposition 9.9 that **S** = $\mathcal{M}$(F). The result that F satisfies the intersection property now follows by Theorem 9.8. □

It is interesting to note that when F is monodependent then the superkey of cardinality k problem can be solved in polynomial time in the size of F [LEVE95b]. This is a corollary of the fact that when F satisfies the split-freeness property then all the keys for R have the same cardinality [LEVE95b].

10 Separatory sets of FDs are monodependent

Several properties of separatory lattices of closed sets are investigated in [DEME92]. In particular when $\mathcal{L}$(F) is separatory, then |$\mathcal{K}$(F)|= 1 and F has a cover whose cardinality is at most (type(R))^2 [BISK91, DEME92]. Herein we show that separatory sets of FDs are monodependent. We also give an example of a set of FDs which is monodependent but not separatory.

Definition 10.1 (Separatory set of FDs) A set of FDs F is *separatory* [DEME92] if it has a cover of the form $\{X_1 \rightarrow A_1, X_2 \rightarrow A_2, \ldots, X_m \rightarrow A_m\}$, where $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_m$. We let RHS(F) denote the set $\{A_1, A_2 \ldots, A_m\}$.

The next lemma is useful in proving the ensuing theorem.

Lemma 10.1 A set of FDs is separatory if and only if it has a canonical cover F of the form $\{X_1 \rightarrow A_1, X_2 \rightarrow A_2, \ldots, X_m \rightarrow A_m\}$, where $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_m$ and $\forall i \in \{1, 2, \ldots, m\}, X_i \cap$ RHS(F) = $\emptyset$.

Proof. We can assume without loss of generality that $\forall$ A $\in$ sch(R), F does not contain distinct FDs of the form X $\rightarrow$ A and Y $\rightarrow$ A. If this were the case then one of X $\rightarrow$ A or Y $\rightarrow$ A is redundant, since either X $\subset$ Y or Y $\subset$ X. Next, let X $\rightarrow$ A be an FD in F.

Claim 1. The FD X $\rightarrow$ A is *not* reduced but Y $\rightarrow$ A $\in F^+$ is reduced, with Y $\subset$ X, if and only if X = YZ, where Z $\neq \emptyset$, Z $\subseteq$ RHS(F) and Y $\cap$ RHS(F) = $\emptyset$. (We observe that |F| > |Z|.)

For the *if part* of the claim let Z = $\{B_1, B_2, \ldots, B_k\}$, with $k > 0$. We use an induction on k to prove the result. For the basis step assume that k = 1 and thus X = YB_1. It follows that W $\subseteq$ Y, where W $\to B_1 \in$ F, since F is separatory, and also that $YB_1 \not\subseteq$ W. Therefore, on using Armstrong's axiom system, $B_1 \in C$(Y) and thus A $\in C$(Y) also. Furthermore, Y $\to$ A is reduced, since Y $\cap$ RHS(F) = $\emptyset$.

For the induction step assume that the result holds when |Z| = k, with k $\geq$ 1; we then need to prove that the result holds when |Z| = k+1. Let V = Y(Z$-B_k$). It follows that W $\subseteq$ V, where W $\to B_k \in$ F, since F is separatory, and also that V $\not\subseteq$ W. Therefore, on using Armstrong's axiom system, $B_k \in C$(V) and thus A $\in C$(V) also. The result follows by inductive hypothesis.

For the *only if part* of the claim consider a nonredundant derivation of Y $\to$ A from F that uses n FDs, with $n > 0$, in the following order: $Y_1 \to B_1, Y_2 \to B_2, \ldots, Y_n \to B_n$ and X $\to$ A, all of which are in F. It follows that $Y_1 \subseteq$ Y, $Y_2 \subseteq YB_1$, ..., $Y_n \subseteq YB_1B_2\ldots B_{n-1}$ and finally X $\subseteq YB_1B_2\ldots B_n$. Therefore, since Y $\subset$ X, we have that X = YZ, where Z $\subseteq$ RHS(F). It remains to show that Y $\cap$ RHS(F) = $\emptyset$. Suppose that this is not the case and hence there is an attribute B $\in$ Y $\cap$ RHS(F). Thus there is an FD W $\to$ B $\in$ F, with W $\subseteq$ Y$-$B, since F is separatory, and Y $\not\subseteq$ W. Therefore, on using Armstrong's axiom system, B $\in C$(Y$-$B), contradicting the fact that Y $\to$ A is reduced. The claim now follows.

From Claim 1 it follows that we can rewrite $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_m$ as $Y_1Z_1 \subseteq Y_2Z_2 \subseteq \ldots \subseteq Y_mZ_m$, where $\forall i \in \{1, 2, \ldots, m\}$, $Y_i \cap$ RHS(F) = $\emptyset$ and $Z_i \subseteq$ RHS(F). The result now follows, since by Claim 1 $\{Y_1 \to A_1, Y_2 \to A_2, \ldots, Y_m \to A_m\}$ is a reduced cover of F, with $Y_1 \subseteq Y_2 \subseteq \ldots \subseteq Y_m$. □

The lattice $\mathcal{L}$(F) of closed sets is said to be *separatory* if $\mathcal{P}$(sch(R)) $-$ $\mathcal{L}$(F) is a *semilattice*, i.e. if it is closed under intersection [GOTT90, LIBK92]. It was shown in [DEME92, Proposition 6.10] that a set of FDs F is separatory if and only if the lattice of closed sets $\mathcal{L}$(F) is separatory. The next result shows that separatory sets of FDs are also monodependent.

Theorem 10.2 If a set of FDs F is separatory, then it is monodependent.

Proof. Assume by Lemma 10.1 that F is canonical and has the form $\{X_1 \to A_1, X_2 \to A_2, \ldots, X_m \to A_m\}$, where $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_m$ and $\forall i \in \{1, 2, \ldots, m\}, X_i \cap$ RHS(F) = $\emptyset$.

By [DEME92, Proposition 6.10] $\mathcal{P}$(sch(R)) $-$ $\mathcal{L}$(F) is a semilattice. Let A $\in$ sch(R). It remains to show that F(A) is a sublattice of $\mathcal{P}$(sch(R)) $-$ $\mathcal{L}$(F), whereupon by Theorem 9.3 F satisfies the intersection property. If there is no FD in F of the form X $\to$ A then the result follows, since F(A) = $\emptyset$. So, let X $\to$ A in F be the reduced FD whose right-hand side is A.

We claim that if Y $\to$ A $\in F^+$ is a nontrivial FD, then X $\subseteq$ Y. Suppose that X $\not\subseteq$ Y and that B $\in$ X$-$Y. By Proposition 4.1 it must be the case that Y $\to$ X $\in F^+$ and thus Y $\to$ B $\in F^+$ is a nontrivial FD. However, by Claim 1 in the proof of Lemma 10.1 it is also true that B $\notin$ RHS(F) and therefore there cannot be a nontrivial FD in F^+ whose right-hand side is B, which leads to a contradiction.

The result that F satisfies the intersection property now follows, since we have that $F(\mathrm{A}) = [\mathrm{X}, \mathrm{sch(R)}-\mathrm{A}]$.

It remains to show that F satisfies the split-freeness property. Suppose to the contrary that there exist cyclic FDs XB $\rightarrow$ A, YA $\rightarrow$ B $\in F^+$, but Y $\rightarrow$ B $\notin F^+$ and (X $\cap$ Y)A $\rightarrow$ B $\notin F^+$. We assume without any loss of generality that YA $\rightarrow$ B is reduced. Now, by Claim 1 in the proof of Lemma 10.1 it follows that there is an FD W $\rightarrow$ A $\in$ F such that B $\notin$ W, due to the fact that XB $\rightarrow$ A $\in F^+$ is a nontrivial FD. Therefore, on using Armstrong's axiom system, WY $\rightarrow$ B $\in F^+$. It follows by the intersection property that Y $\rightarrow$ B $\in F^+$, since A $\notin$ W. Hence YA $\rightarrow$ B is not reduced leading to a contradiction of our assumption. The result that F satisfies the split-freeness property follows as required. □

As the following example shows a set of FDs may be monodependent but not separatory.

Example 10.1 Let F = {A $\rightarrow$ B, D $\rightarrow$ E}, with sch(R) = {A, B, D, E}. It can easily be verified that F is monodependent but not separatory.

11 Concluding Remarks

Monodependence is a desirable property of sets of FDs when assuming that relations may be incomplete. We have investigated the structure of the lattice of closed sets $\mathcal{L}$(F) when F is monodependent. As a consequence of this investigation we have shown that monodependent sets of FDs give rise to several desirable properties. Moreover, several difficult problems in relational database theory become tractable when F is monodependent. The connection between lattice theory and relational database theory is important, since it provides us with additional insight into the semantics of data dependencies such as FDs. A lattice-theoretic investigation of MVDs was carried out in [DAY93]. We conclude by giving a brief summary of the main results.

Assume that F satisfies the intersection property. In Theorem 8.2 we show that $\mathcal{L}$(F) is exchange if and only if the cardinality of all the nonempty equivalence classes of F is maximal. On the other hand, in Theorem 8.3 we show that $\mathcal{L}$(F) is antiexchange if and only if the cardinality of all the nonempty equivalence classes of F is minimal, i.e. it is one.

In Theorem 9.3 we give a characterisation of the intersection property in terms of the existence of certain distributive sublattices of $\mathcal{L}$(F). In Corollary 9.5 we show that the cardinality of $\mathcal{M}$(F) is at most m, where $m = \binom{\mathrm{type(R)}}{\mathrm{type(R)}-2}$. Thus an Armstrong relation having $m+1$ tuples can be generated. In Corollary 9.7 we show that the prime attribute problem can be solved in polynomial time in the size of F. In Theorem 9.8 we show that F satisfies the intersection property if and only if the cardinality of each element in $\mathcal{M}$(F) is greater than or equal to type(R)$-$2. Using this result we are able to show in Theorem 9.11 that the superkey of cardinality k problem is still NP-complete, when F is restricted to be a set of FDs that satisfies the intersection property. Finally, in Theorem 10.2 we show that separatory sets of FDs are monodependent.

References

[ARMS74] W.W. Armstrong, Dependency structures of data base relationships. In: *Proceedings of the IFIP Congress*, Stockholm, pp. 580-583, 1974.

[ATZE86] P. Atzeni and N.M. Morfuni, Functional dependencies and constraints on null values in database relations. *Information and Control* **70**, (1986), 1-31.

[ATZE93] P. Atzeni and V. De Antonellis, *Relational Database Theory.* Benjamin/Cummings, Redwood City, CA., 1993.

[BEER79] C. Beeri and P.A. Bernstein, Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems* **4**, (1979), 30-59.

[BEER84] C. Beeri, M. Dowd, R. Fagin and R. Statman, On the structure of Armstrong relations for functional dependencies. *Journal of the ACM* **31**, (1984), 30-46.

[BEER86] C. Beeri and M. Kifer, Elimination of intersection anomalies for database schemes. *Journal of the ACM* **33**, (1986), 423-450.

[BISK91] J. Biskup, J. Demetrovics, L.O. Libkin and I.B. Muchnik, On relational database schemes having unique minimal key. *Journal of Information Processing Cybernetics* **27**, (1991), 217-225.

[BURO87] G. Burosch, J. Demetrovics and G.O.H. Katona, The poset of closures as a model of changing databases. *Order* **4**, (1987), 127-142.

[CODD79] E.F. Codd, Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems* **4**, (1979), 397-434.

[COSM90] S.S. Cosmadakis, P.C. Kanellakis and M.Y. Vardi, Polynomial-time implication problems for unary inclusion dependencies. *Journal of the ACM* **37**, (1990), 15-46.

[DAVE90] B.A. Davey and H.A. Priestly, *Introduction to Lattices and Order.* Cambridge University Press, Cambridge, U.K., 1990.

[DAY93] A. Day, A lattice interpretation of database dependencies. In: *Semantics of Programming Languages and Model Theory*, M. Droste and Y. Gurevich (Eds.), Gordon and Breach, pp. 305-325, 1993.

[DEME87] J. Demetrovics and V.D. Thi, Keys, antikeys and prime attributes. *Annales Univ. Sci. Budapest, Sect. Comp.* **8**, (1987), 35-52.

[DEME88] J. Demetrovics and V.D. Thi, Relations and minimal keys. *Acta Cybernetica* **8**, (1988), 279-285.

[DEME92] J. Demetrovics, L. Libkin and I.B. Muchnik, Functional dependencies in relational databases: A lattice point of view. *Discrete Applied Mathematics* **40**, (1992), 155-185.

[DEME93] J. Demetrovics and G.O.H. Katona, A survey of some combinatorial results concerning functional dependencies in database relations. *Annals of Mathematics and Artificial Intelligence* **7**, (1993), 63-82.

[GARE79] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman, New York, 1979.

[GOTT90] G. Gottlob and L. Libkin, Investigations on Arsmtrong relations, dependency inference and excluded functional dependencies. *Acta Cybernetica* **9**, (1990), 385-402.

[GRAT78] G. Grätzer, *General Lattice Theory.* Academic Press, New York, 1978.

[JAMI85] R.E. Jamison, The theory of convex geometries. *Geometriae Dedicata* **19**, (1985), 247-270.

[LEVE94] M. Levene and G. Loizou, The additivity problem for functional dependencies in incomplete relations. Research Note RN/94/41, Department of Computer Science, University College London, 1994. To appear in *Acta Informatica.*

[LEVE95a] M. Levene and G. Loizou, The additivity problem for data dependencies in incomplete relational databases. In: *Proceedings of the Workshop on Semantics in Databases*, Prague, pp. 49-62, 1995.

[LEVE95b] M. Levene and G. Loizou, Database design for incomplete relations. Research Note RN/95/18, Department of Computer Science, University College London, 1995. Submitted for publication.

[LIBK92] L. Libkin and I. Muchnik, Separatory sublattices and subsemilattices. *Studia Scientiarum Mathematicarum Hungarica* **27**, (1992), 471-477.

[LIEN82] Y.E. Lien, On the equivalence of database models. *Journal of the ACM* **29**, (1982), 333-362.

[LUCC78] C.L. Lucchesi and S.L. Osborn, Candidate keys for relations. *Journal of Computer and System Sciences* **17**, (1978), 270-279.

[MANN83] H. Mannila and K-J. Räihä, On the relationship of minimum and optimum covers for a set of functional dependencies. *Acta Informatica* **20**, (1983), 143-158.

[MANN86] H. Mannila and K-J. Räihä, Design by example: an application of Armstrong relations. *Journal of Computer and System Sciences* **33**, (1986), 126-141.

[MANN89] H. Mannila and K.-J. Räihä, Practical algorithms for finding prime attributes and testing normal forms. In: *Proceedings of the ACM Symposium on Principle of Database Systems*, Philadelphia, PA., pp. 128-133, 1989.

[MANN92] H. Mannila and K-J. Räihä, *The Design of Relational Databases.* Addison-Wesley, Reading, MA., 1992.

[MAIE80] D. Maier, Minimum covers in the relational database model. *Journal of the ACM* **27**, (1980), 664-674.

[SCIO81] E. Sciore, Real world mvd's. In: *Proceedings of ACM SIGMOD Conference on Management of Data*, Ann Arbor, pp. 121-132, 1981.

[THI86] V.D. Thi, Minimal keys and antikeys. *Acta Cybernetica* **7**, (1986), 361-371.

[ULLM88] J.D. Ullman, *Principles of Database and Knowledge-Base Systems, Vol. I.* Computer Science Press, Rockville, MD., 1988.

[WILD89] M. Wild, Optimal implicational bases for finite modular lattices. Preprint 1211, Technische Hochschule Darmstadt, Germany, 1989.

[WILD95] M. Wild, Computations with finite closures systems and implications. In: *Proceedings of the International Conference on Computing and Combinatoric Xian, China, pp. 111-120, 1995.*

Received September, 1995

Acta Cybernetica 12 (1995) 209-216.

On a tour construction heuristic for the asymmetric TSP

I. Bartalos * T. Dudás† B. Imreh *

Abstract

In this paper we deal with a new tour construction procedure for the asymmetric traveling salesman problem. This heuristic is based on a new patching operation which joins three subtours together. Regarding the efficiency of this procedure, we present an empirical analysis.

It is well-known that the assignment problem is a relaxation of the traveling salesman problem (TSP). Thus, if the optimal assignment is a tour, then it is also an optimal solution of the TSP. Otherwise it consists of disjoint cycles. For some special cases of the TSP, these cycles can be patched into an optimal tour. The first algorithm based on this technique was presented by P. C. Gilmore and R. E. Gomory in [1]. Their idea was involved in several procedures solving different special TSP models. A nice discussion of the well-solved cases can be found in [3].

For the general TSP there is no effective procedure to convert the optimal assignment into an optimal tour. Nevertheless as the computational experiments of E. Balas and P. Toth (see [3]) show, the lower bound resulting from the assignment problem is often very tight. On the other hand, the number of the cycles of the optimal assignment is not large in general. These facts suggest that a suitable subtour patching method may result in a good TSP heuristic. The first such heuristic for the asymmetric TSP was presented by R. M. Karp [5] and a similar one was given in [3]. In both cases the method converts the optimal assignment into a tour by a sequence of patching operations, each of which joins two cycles together. Our procedure is based on such a patching operation which joins three cycles together if the number of the cycles of the optimal assignment is not large. Algorithms joining four or more cycles in each step have too high complexity.

As far as the number of the cycles is concerned, it is known that if we choose a permutation of $\{1, \ldots, n\}$ at random, then the expected number of its cycles is $\log(n)$ (see e.g. [4]). We can, however, restrict our consideration for permutations without a fixpoint. Indeed, without loss of generality we may assume that the optimal assignment does not contain diagonal elements. Such an assignment can be achieved by choosing suitably large coefficients in the diagonal of the cost matrix.

*Department of Informatics, József Attila University, P.O. Box 652, H6701 Szeged, Hungary
†Institute of Mathematics, Technical University of Graz, Kopernikusgasse 24, A-8010 Graz, Austria.

The expected number of cycles in a randomly chosen permutation without a fixpoint has not been calculated as yet. Here we prove that the approximate value of this number is still $\log(n)$.

To start with, let us denote by R_n the set of permutations without a fixpoint on the set $\{1,\ldots,n\}$ and let $|R_n| = r_n$. It is known (see [2] p.10) that

$$r_n = n!(1 - \frac{1}{1!} + \ldots + (-1)^n \frac{1}{n!}) \approx \frac{n!}{e}.$$

Let $2 \le k \le n-2$ be an arbitrary fixed integer and $i \in \{1,\ldots,n\}$ where $n \ge 5$. Let us count those permutations of R_n in which i is contained in a cycle of length k. There are $\binom{n-1}{k-1}$ possible ways to choose the elements of this cycle and $(k-1)!$ ways to order them. The number of the permutations without a fixpoint of the remaining $n-k$ elements is r_{n-k}. Therefore, the number we seek is

$$\binom{n-1}{k-1}(k-1)! r_{n-k}\,.$$

It is obvious that the number of the permutations in which i is contained in a cycle of length n is $(n-1)!$.

Now let us consider R_n as a sample space in which each permutation is assigned a probability $1/r_n$. For any $i \in \{1,\ldots n\}$ and $k \in \{2,3\ldots,n-2,n\}$, let us denote by $\xi_i^{(k)}$ the random variable on R_n for which

$$\xi_i^{(k)} = \begin{cases} 1 & \text{if } i \text{ is contained in a } k\text{-cycle,} \\ 0 & \text{otherwise.} \end{cases}$$

Using the numbers determined above, we obtain

$$E(\xi_i^{(k)}) = \binom{n-1}{k-1}(k-1)!\frac{r_{n-k}}{r_n} \quad \text{if} \quad 2 \le k \le n-2 \quad \text{and}$$

$$E(\xi_i^{(n)}) = (n-1)!\frac{1}{r_n}\,.$$

Now $\xi_1^{(k)} + \ldots + \xi_n^{(k)}$ is the number of points which are contained in k-cycles, and $\eta_k = \frac{1}{k}(\xi_1^{(k)} + \ldots + \xi_n^{(k)})$ is the number of k-cycles. The expected value of η_k is

$$E(\eta_k) = \frac{n}{k}\binom{n-1}{k-1}(k-1)!\frac{r_{n-k}}{r_n} = \frac{n!}{k(n-k)!}\frac{r_{n-k}}{r_n} \quad \text{if} \quad 2 \le k \le n-2 \quad \text{and}$$

$$E(\eta_n) = \frac{(n-1)!}{r_n}\,.$$

Then $\eta_2 + \eta_3 \ldots + \eta_{n-2} + \eta_n$ is the number of cycles and for the expected value u_n of this number, we obtain

$$u_n = \frac{1}{r_n}\left((n-1)! + \sum_{k=2}^{n-2} \frac{n! r_{n-k}}{k(n-k)!}\right).$$

Now substituting r_{n-k} and exchanging the order of the summation, we get that u_n is equal to

$$\frac{1}{r_n}\left((n-1)! + \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-i} \frac{n!}{k}\right) =$$

$$\frac{n!}{r_n}\left(\frac{1}{n} + \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-2} \frac{1}{k} - \sum_{i=3}^{n-2} \frac{(-1)^i}{i!} \sum_{k=n-i+1}^{n-2} \frac{1}{k}\right).$$

Let

$$w_i = \frac{1}{i!} \sum_{k=n-i+1}^{n-2} \frac{1}{k}, \quad i = 3, \ldots, n-2.$$

Then

$$u_n = \frac{n!}{r_n}\left(\frac{1}{n} + \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-2} \frac{1}{k} + \sum_{i=3}^{n-2} (-1)^{i+1} w_i\right).$$

It is easy to see that $w_3 > \ldots > w_{n-2} > 0$, and so,

$$0 < \sum_{i=3}^{n-2} (-1)^{i+1} w_i \leq w_3.$$

On the other hand,

$$\frac{1}{n} + \frac{1}{3!}\frac{1}{n-2} < \frac{7}{6(n-2)}.$$

Therefore,

$$\frac{n!}{r_n}\left(\sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-2} \frac{1}{k}\right) < u_n < \frac{n!}{r_n}\left(\frac{7}{6(n-2)} + \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-2} \frac{1}{k}\right).$$

If n is large enough, then

$$\frac{n!}{r_n} \approx e, \qquad \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \approx e^{-1}, \qquad \frac{7e}{6(n-2)} + \sum_{k=2}^{n-2} \frac{1}{k} \leq \log(n-2),$$

and so, $\log(n-1) - 1 \leq u_n \leq \log(n-2)$.

Now we recall the definition of the patching operation (see [3]). For this reason let us consider an asymmetric TSP of n cities with cost matrix $\mathbf{C}$ and let φ denote an optimal assignment which is not a tour. For the sake of notation simplicity, we shall identify every cycle with the set of its cities. Assuming that φ has no fixpoint, let i and j be two cities that occur in two distinct cycles U and V. Then deleting the arcs $(i,\varphi(i))$, $(j,\varphi(j))$ and inserting the arcs $(i,\varphi(j))$, $(j,\varphi(i))$, we join U and V into a new cycle, and thus obtain a new assignment $\bar{\varphi}$. This operation is called the (i,j)-*patching operation*. For the cost of the new assignment, we get

$$z(\bar{\varphi}) = z(\varphi) + c_{i\varphi(j)} + c_{j\varphi(i)} - c_{i\varphi(i)} - c_{j\varphi(j)}\,.$$

The difference $z(\bar{\varphi}) - z(\varphi) = c_{i\varphi(j)} + c_{j\varphi(i)} - c_{i\varphi(i)} - c_{j\varphi(j)}$ is called the *patching cost of the* (i,j)-*patching operation*. The minimal patching cost with respect to U and V is

$$\Delta(\varphi,U,V) = \min\{c_{r\varphi(s)} + c_{s\varphi(r)} - c_{r\varphi(r)} - c_{s\varphi(s)} : r \in U,\, s \in V\}\,.$$

This means that the cycles U and V can be joined into a new cycle with cost $\Delta(\varphi,U,V)$, but they cannot be joined with a lower cost by any (i,j)-patching operation. Therefore we say that the *2-patching cost of the cycles U and V* is $\Delta(\varphi,U,V)$.

Now we are ready to present the algorithm developed in [3].

The 2-patching algorithm

Step 1. Determine an optimal assignment φ.

Step 2. If the current φ is a cyclic permutation then terminate.
Otherwise go to Step 3.

Step 3. Choose two cycles U and V of φ such that $|U|$ and $|V|$ are maximal.
Calculate $\Delta(\varphi,U,V)$ and let $i \in U$, $j \in V$ such cities for which the patching cost is $\Delta(\varphi,U,V)$. Perform the (i,j)-patching operation and consider the new assignment as the current φ. Return to Step 2.

Extending the patching idea for three cycles, we can define the (i,j,k)-*patching operation* as follows.

Let i,j,k be three cities which occur in three distinct cycles U,V,W. Then deleting the arcs $(i,\varphi(i))$, $(j,\varphi(j))$, $(k,\varphi(k))$ and inserting the arcs $(i,\varphi(j))$, $(j,\varphi(k))$, $(k,\varphi(i))$, we join U, V and W into a new cycle. The patching cost of this operation is

$$c_{i\varphi(j)} + c_{j\varphi(k)} + c_{k\varphi(i)} - c_{i\varphi(i)} - c_{j\varphi(j)} - c_{k\varphi(k)}\,.$$

The minimal patching cost with respect to U,V,W is

$$\Theta(\varphi,U,V,W) = \min\{c_{r\varphi(s)}+c_{s\varphi(t)}+c_{t\varphi(r)}-c_{r\varphi(r)}-c_{s\varphi(s)}-c_{t\varphi(t)}: r \in U, s \in V,\ t \in W\}.$$

Then there are cities $i \in U$, $j \in V$, $k \in W$ such that the cycles U, V, W can be joined into a cycle with cost $\Theta(\varphi, U, V, W)$. This cost is called the *3-patching cost of the cycles* U, V, W.

Now we present a procedure based on the introduced (i, j, k)-patching operations. Since $9 \sim \log(8100)$, the expected value of the disjoint cycles is not greater than 9 under the problem size 8100. In practical point of view this limit of problem size is enough large, and so, our procedure uses (i, j, k)-patching operations while the number of the disjoint cycles is not greater than 9. For the extreme cases, when the number of the disjoint cycles is greater than 9, we apply an additional step (*Step 3*) to pair the small cycles with the large ones and to join them by suitable (i, j)-patching operations.

The 3-patching algorithm

Step 1. Determine an optimal assignment φ.

Step 2. If the current φ is a cyclic permutation then terminate. Otherwise go to Step 3.

Step 3. Let m denote the number of the cycles of φ. If $m \leq 9$ then go to Step 4. Otherwise order the cycles with respect to the number of vertices belonging to them. Let $U_1, \ldots, U_m$ denote the sequence of the cycles in increasing order. Calculate the 2-patching cost d_{rs} of the cycles U_r and U_{m-l+s} for all $1 \leq r \leq l$ and $1 \leq s \leq l$, where $l = [m/2]$. Solve the assignment problem of type $l \times l$ with the cost matrix $\mathbf{D} = (d_{rs})$. Let τ denote an optimal assignment. For all $1 \leq r \leq l$, join the cycles U_r and $U_{m-l+\tau(r)}$ with patching cost $d_{r\tau(r)}$, using a suitable (i, j)-patching operation. Consider the assignment obtained after the l joins as the current assignment φ and repeat Step 3.

Step 4. If $m = 2$, then determine the 2-patching cost of the two cycles of φ, join them with a suitable (i, j)-patching operation and terminate. If $m > 2$, then choose three cycles U, V, W of φ for which $\Theta(\varphi, U, V, W)$ is minimal. Determine three cities $i \in U$, $j \in V$, $k \in W$ with the 3-patching cost $\Theta(\varphi, U, V, W)$. Perform the (i, j, k)-patching operation and consider the new assignment as the current φ. If φ is a cyclic permutation then terminate. Otherwise repeat Step 4 with the current φ.

In order to efficiently find the three cycles having minimal 3-patching cost required in Step 4, we maintain a 3-dimensional array of the values $\Theta(\varphi, U, V, W)$. To set up this array we firstly need $O(n^3)$ operations, but during the iterations one can compute the new array from the previous one easily.

In both patching procedures above the starting point is an optimal assignment which can be computed by the Hungarian method in $O(n^3)$ steps. This method starts with an independent set of zeroes in the reduced cost matrix. These independent zeroes can be determined randomly. In general, different independent sets of zeroes result in different optimal assignments. Using this observation and executing the procedure k times, we can obtain k distinct heuristic solutions and we choose the best of them.

To test the efficiency of our procedure we performed the following computational experiment on a 33MHz 486 machine. We randomly generated 100-100 problems under $n = 100$, $n = 150$, $n = 200$ $n = 250$, respectively, with costs independently drawn from a uniform distribution of the integers over the interval $[0, 100]$. We applied some classical tour construction heuristics and three versions of both the 2-patching and the 3- patching methods to obtain heuristic solutions for the generated problems. The difference among the three versions appears in the number k of executions under a randomly choosen independent set of zeroes. We worked with the values $k = 1, 3, 5$. Simultaneously, we solved these problems by a branch and bound procedure, using the best heuristic solution as the best known feasible solution. Regarding the classical heuristics we applied the all cities versions of the nearest addition, nearest insertion and farthest insertion algorithms, and our cheapest insertion procedure started from a shortest two city subtour. The results of our computational experiments are reported in Table 1. Here the first column gives the averages of the ratios $z(\textit{heuristic solution})/z(\textit{optimal solution})$, the second column contains the averages of the run-times in seconds and the third column shows how many times the suitable heuristic gave the best solution among the ones provided by all the 11 heuristics considered.

	n = 100 average ratio	average sec.	best value	n = 150 average ratio	average sec.	best value	n = 200 average ratio	average sec.	best value	n = 250 average ratio	average sec.	best value
B&B	1.000	40.78	-	1.000	87.69	-	1.000	194.1	-	1.000	320.9	-
3- patching k=5	1.054	19.53	88	1.056	58.55	81	1.052	190.2	82	1.059	370.8	80
3- patching k=3	1.061	11.60	70	1.063	34.93	67	1.061	122.0	54	1.078	218.6	48
3- patching k=1	1.069	3.84	55	1.096	11.90	39	1.094	39.8	25	1.134	72.6	24
2- patching k=5	1.090	11.14	33	1.082	29.70	38	1.069	88.4	40	1.101	158.3	37
2- patching k=3	1.092	6.69	28	1.097	17.92	26	1.085	53.1	27	1.119	94.8	23
2- patching k=1	1.108	2.21	21	1.127	6.04	15	1.127	17.7	13	1.177	31.5	12
cheapest insertion	4.654	7.77	0	6.794	37.48	0	9.934	89.7	0	18.11	175.4	0
nearest insertion	4.392	9.19	0	7.047	43.66	0	11.39	104.6	0	18.60	205.1	0
farthest insertion	4.534	8.76	0	7.110	43.71	0	11.39	104.6	0	18.60	205.3	0
nearest addition	18.43	7.09	0	33.84	32.72	0	57.51	77.0	0	98.43	149.7	0

Table 1

According to the obtained results, both patching algorithms appear to be better than the investigated insertion procedures. Moreover, the 3-patching method seems to provide a good approximate solution almost independently of the problem size. Due to the very good starting feasible solution, the branch and bound method also turns out to be rather effective.

Acknowledgment. The authors would like to thank R. E. Burkard for his valuable comments.

References

[1] P. C. Gilmore and R. E. Gomory, Sequencing a one state-variable machine: a solvable case of the traveling salesman problem, *Oper. Res.* **12**, 655-679 (1964).

[2] M. Hall, *Combinatorial Theory,* Blaisdell, Waltham, 1967.

[3] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, 1985

[4] L. Lovász, *Combinatorial Problems and Exercises*, Akadémiai Kiadó, Budapest, 1979.

[5] R. M. Karp, A patching algorithm for the nonsymmetric traveling salesman problem, *SIAM J. Comput.*, **8**, 561-575 (1979).

Received April, 1995

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 1995 Numbers 1–2 of Volume 12 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-6720 Szeged, P.O.Box 652, Hungary. Tel.: (36)-(62)-311-184, Fax:(36)-(62)-312-292.

CONTENTS

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János
A kézirat a nyomdába érkezett: 1995. december
Terjedelem: 7,12 (B/5) ív