Volume 9 Number 4

ACTA CYBERNETICA

Szeged, 1990

Information for authors: Acta Cybernetica publishes only original papers in English in the field of computer sciences. Review papers are accepted only exceptionally. Manuscripts should be sent in triplicate to one of the Editors. The manuscript must be typed double-spaced on one side of the paper only. For the form of references, see one of the articles previously published in the journal. A list of special symbols should be supplied by the authors.

Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1990

Recognizable sets of finite bilabelled transition systems

A. Arnold
Laboratoire d'Informatique *
Université Bordeaux I

Abstract

In this note we propose a definition of a notion of automaton recognizing finite bilabelled transition systems, i.e. finite directed graphs with labels attached to both vertices and edges. The family of recognizable sets is a boolean algebra. Moreover every recognizable set contains the images and the inverse images of each of its element under surjective homomorphisms.

1 Introduction

Recognizable sets of finite words and of finite trees are defined by mean of automata: a recognizable set consists in all the elements recognized by such an automaton. Since words are special kinds of trees, the notion of tree automaton is an extension of the notion of word automaton. But trees are special kinds of directed acyclic graphs (*dags*, for short) and indeed a notion of dag automaton was recently introduced [3], which is an extension of the notion of tree automaton.

Since dags are special kinds of graphs, one can think of a notion of graph automaton which is an extension of the notion of dag automaton. However there is a major distinction between dags and graphs: when an automaton reads in a dag it goes from vertices to vertices along the directed edges, and, because of acyclicity, it never reads in twice the same vertex; therefore it is possible to assign a unique state of the automaton to every newly read in vertex according to the states assigned to previous vertices as specified by the transition function of the automaton. For graphs, the situation is different, since a vertex can be read in several times and then the state assigned to a vertex can change during the computation of the automaton.

But it is possible to interpret the results of Büchi on infinite words [4], and of Rabin on infinite trees [8], as an intuitive support to the thesis that there is a close connection between the notion of recognizability and the notion of definability by some monadic second order logic (see for instance [5]). Therefore one can investigate for a characterization of the set of graphs which are models of a given monadic second order formula in terms of automata.

*Unité de Recherche associée au Centre National de la Recherche Scientifique n° 1304

Since Branching Time Temporal Logics as well as the μ-calculus [6] are special cases of monadic second order logics which are used to express properties of processes, and since processes are usually represented by finite transition systems, one can expect to get a notion of automaton which recognizes those transition systems satisfying some given temporal properties. Indeed, the μ-calculus already gives a precious hint on what a graph automaton could be: when computing the value of a formula over a graph, the boolean value of each subformula is computed at every vertex of the graph; hence one can see every vector of boolean values of these subformulas as a possible state of the automaton. This set of states is naturally ordered by a partial order. This led us to consider as the set of states of a graph automaton a finite partially ordered set. Provided this set has a minimal element and the transition function is monotonic, one can define a *computation* of such an automaton in the following way: initially the minimal state is assigned to every vertex; due to the monotonicity of the transition function, when the state assigned to a vertex has to change it can only increase; when the states assigned to the vertices cannot be increased, the computation ends. In other words the assigment reached at the end of the computation is the least fixed-point of some mapping associated with the transition function. Let us remark that in the case of infinite trees, a *run* is sometimes defined as an assigment of states to nodes which satisfies some relations determined by the transition relation of the automaton.

In the first part we consider the simple case of complete deterministic bilabelled transition systems. They are defined as a set of vertices with a label attached to each vertex and with a mapping from the set of vertices into itself associated with each element of some alphabet. We define our notion of automaton in this simple case. In the second part, we consider the more general case where, with each element of the alphabet is associated a mapping from the set of vertices into its powerset. In the third part we define the product of automata and in then we define the acceptance criteria for these automata and give some properties of recognizable sets: the family of recognizable sets is a boolean algebra and each recognizable set is closed under surjective homomorphisms and inverse surjective homomorphisms. Therefore, every recognizable set is a union of *fibers*, where a fiber is a set of inverse surjective homomorphic images of one bilabelled transition system, and we show that a fiber is a recognizable set. The automata previously defined are deterministic and the sets they recognized are called "deterministically recognizable"; we define also "nondeterministically recognizable sets" as beeing the projections of deterministically recognizable ones.

This note consists mainly in definitions although some elementary open questions remain to be answered. But the main question raised by this definition of recognizable sets is the following: it is well known that, in the cases of words and of trees, there exist special kinds of grammars (the regular grammars) which generates exactly the recognizable sets; thus, among the large number of kinds of graphs grammars already introduced in the litterature, do there exist some kinds of them which can generate exactly the determistically or non deterministically recognizable sets of bilabelled transition systems.

2 A simple case

2.1 Complete deterministic bilabelled transition systems

A complete deterministic bilabelled transition system over an alphabet A and a set L of labels is a tuple $S = < V, \gamma, \lambda >$ where

- V is a finite set of vertices,

- γ is a mapping from $A \times V$ into V; $\gamma(a, v)$ is the unique vertex of V which is reachable from v by an edge labelled by a,
- λ is a mapping from V into L.

2.2 Deterministic cdbts automata

A deterministic cdbts automaton is a tuple $\mathcal{S} =< A, L, Q, q_0, \delta >$ where

- A is an alphabet and L is a set of labels,
- Q is a finite partially ordered set of states, and q_0 is its minimal element,
- δ, the transition function, is a monotonic mapping from $L \times Q^A$, ordered componentwise, into Q.

Given a cdbts $\mathcal{S} =< V, \gamma, \lambda >$ over A and L and a deterministic automaton $\mathcal{A} = < A, L, Q, q_0, \delta >$, the set $\mathbf{A}$ of *assignments* is the ordered set Q^V, the minimal element of which is $\{q_0\}^V$. The mapping δ is extended into a monotonic mapping $\delta_{\mathcal{S}}$ from $\mathbf{A}$ into $\mathbf{A}$ defined as follows:

Let α be an element of $\mathbf{A} = Q^V$; let v be an element of V; its *environment under* α is the tuple $\mathrm{env}^{\alpha}_{\mathcal{S}}(v) =< \lambda(v), \alpha(\gamma(a_1, v)), \ldots, \alpha(\gamma(a_n, v)) >$, where $A = \{a_1, \ldots, a_n\}$, which is an element of $L \times Q^A$ and $\delta(\mathrm{env}^{\alpha}_{\mathcal{S}}(v))$ is an element of Q. Then $\delta_{\mathcal{S}}(\alpha)$ is the element of $\mathbf{A}$ defined by $\delta_{\mathcal{S}}(\alpha)(v) = \delta(\mathrm{env}^{\alpha}_{\mathcal{S}}(v))$.

Obviously the mapping $\delta_{\mathcal{S}}$ is monotononic, and since $\mathbf{A}$ has a minimal element, $\delta_{\mathcal{S}}$ has a least fixed point in $\mathbf{A}$ which will be denoted by $\mu\delta_{\mathcal{S}}$.

The following example mainly shows in which sense deterministic word automata are a special kind of cdbts automata. This construction can easily be extended to bottom-up deterministic tree automata.

Example 1. Let us consider a usual deterministic word automaton $\mathcal{A}$ over an alphabet L with Q as a set of states.

With a word $u = l_1 \ldots l_n$ we associate the cdbts $\mathcal{S}_u$ over a single-letter alphabet and with $L \cup \{\vdash\}$ as set of labels defined as follows:

- the set of vertices is $\{0, 1, \ldots, n\}$,
- there is an edge from i to $i - 1$ (with $0 - 1 = 0$)
- 0 is labelled with $\vdash$, and i, for $1 \leq i \leq n$ is labelled with l_i,

Then we define a cdbts automaton over a single-letter alphabet and the set of labels $L \cup \{\vdash\}$ by

- the set of states is $Q \cup \{\perp\}$, where $\perp$ is less than any other element,
- the transition function is defined by
  - $\delta(\vdash, q)$ is the initial state of Q, for every q in $Q \cup \{\perp\}$,
  - $\delta(l, \perp) =\perp$ for every l in L,
  - $\delta(l, q)$ is the state obtained by applying the transition function of the deterministic word automaton to $l \in L$ and $q \in Q$.

It is easy to see that in the assignment which is the least fixed point of $\delta_{\mathcal{S}_u}$, the initial state of $\mathcal{A}$ is assigned to the vertex 0 and the state reached by $\mathcal{A}$ reading $l_1 \ldots l_i$ and starting in the initial state is assigned to the vertex i. □

2.3 Homomorphisms of cdbts

Let $S_1 =< V_1, \gamma_1, \lambda_1 >$ and $S_2 =< V_2, \gamma_2, \lambda_2 >$ be two cdbts. An *homomorphism* from S_1 into S_2 is a mapping $h : V_1 \longrightarrow V_2$ such that $h(\gamma_1(a, v)) = \gamma_2(a, h(v))$ and $\lambda_1(v) = \lambda_2(h(v))$.

If $\mathcal{A} =< A, L, Q, q_0, \delta >$ is a cdbts automaton over A and L, let $\mu\delta_{S_1}$ and $\mu\delta_{S_2}$ be the least fixed points of δ_{S_1} and δ_{S_2}.

For every assignment β in Q^{V_2}, $\beta \circ h$, defined by

$$\beta \circ h(v) = \beta(h(v))$$

is an assignment in Q^{V_1}. In particular we have:

Lemma 2.1 *If h is a homomorphism from S_1 into S_2, and if β is an assignment in Q^{V_2}, then for any v in V_1,*

$$\mathrm{env}^{\beta}_{S_2}(h(v)) = \mathrm{env}^{\beta \circ h}_{S_1}(v)$$

Proof By definition of env we have

$$\mathrm{env}^{\beta}_{S_2}(h(v)) =< \lambda_2(h(v)), \beta(\gamma_2(a_1, h(v))), \ldots, \beta(\gamma_2(a_n, h(v))) >$$

and

$$\mathrm{env}^{\beta \circ h}_{S_1} =< \lambda_1(v), \beta(h(\gamma_1(a_1, v))), \ldots, \beta(h(\gamma_1(a_n, v))) >$$

Since h is an homomorphism, the two right-hand sides of these equations are equal, hence the result. □

It follows

Lemma 2.2 *If h is an homomorphism from S_1 into S_2, and if β is an assignment in Q^{V_2}, then*

$$\delta_{S_1}(\beta \circ h) = \delta_{S_2}(\beta) \circ h$$

Proof For any v in V_1 we have

$$\delta_{S_1}(\beta \circ h)(v) = \mathrm{env}^{\beta \circ h}_{S_1}(v)$$

and

$$\delta_{S_2}(\beta) \circ h(v) = \delta_{S_2}(\beta)(h(v)) = \mathrm{env}^{\beta}_{S_2}(h(v))$$

and the result immediately follows from Lemma 2.1 □

Then we get

Theorem 2.1 *If h is an homomorphism from S_1 into S_2, then*

$$\mu\delta_{S_1} = \mu\delta_{S_2} \circ h$$

Proof Let us denote by $\perp_k$, for $k = 1, 2$ the least mapping in Q^{V_k} which associates the minimal element q_0 of Q with every vertex v of V_k.

α, the least fixed point of δ_{S_1}, is the limit of the increasing sequence $(\alpha_i)_{i\geq 0}$ defined by $\alpha_0 = \perp_1$ and $\alpha_{i+1} = \delta_{S_1}(\alpha_i)$. Similarly, β, the least fixed point of δ_{S_2}, is the limit of the increasing sequence $(\beta_i)_{i\geq 0}$ defined by $\beta_0 = \perp_2$ and $\beta_{i+1} = \delta_{S_2}(\beta_i)$.

We prove by induction that $\alpha_i = \beta_i \circ h$.

Obviously

$$\beta_0 \circ h = \perp_2 \circ h = \perp_1 = \alpha_0$$

and, by Lemma 2.2,

$$\beta_{i+1} \circ h = \delta_{S_2}(\beta_i) \circ h = \delta_{S_1}(\beta_i \circ h)$$

which is equal, by induction hypothesis, to

$$\delta_{S_1}(\alpha_i) = \alpha_{i+1}$$

□

3 A more general case

We want to extend the previous definition of automata to the case of non deterministic bilabelled transition systems (bts). In this case $\gamma(a, v)$ is no longer a single vertice, but a set (possibly empty) of vertices. Hence the following definition:

3.1 Non deterministic bilabelled transition systems

A non deterministic bilabelled transition system over an alphabet A and a set L of labels is a tuple $S = < V, \gamma, \lambda >$ where

- V is a finite set of vertices,
- γ is a mapping from $A \times V$ into $\wp(V)$; $\gamma(a, v)$ is the set of vertices of V which are reachable from v by an edge labelled by a,
- λ is a mapping from V into L.

3.2 The powerset of states

If Q is a partially ordered set of states with a minimal element $\perp$, we can still define the image of a set of vertices under an assignment in Q^V, but, for the mapping δ being monotonic we need some partial order on $\wp(Q)$. We choose to use the so called *Egli-Milner preorder* [7] defined by

$$\begin{aligned} X \sqsubseteq Y \text{ iff } \forall x \in X, \exists y \in Y : x &\leq y \\ \text{and } \forall y \in Y, \exists x \in X : x &\leq y. \end{aligned}$$

This preorder has the following properties:

Proposition 3.1

- *The empty set is not comparable with any non empty set.*
- *The set $\{\perp\}$ is less than any non empty set.*
- *If α and β are two assignments in Q^V, and if U is a set of vertices, then $\alpha \leq \beta$ implies $\alpha(U) \sqsubseteq \beta(U)$*

3.3 Deterministic bts automata

A deterministic bts automaton is a tuple $\mathcal{S} =< A, L, Q, q_0, \delta >$ where

- A is an alphabet and L is a set of labels,
- Q is a finite partially ordered set of states, and q_0 is its minimal element, $\wp(Q)$ being ordered by the Egli-Milner preorder,
- δ, the transition function, is a monotonic mapping from $L \times \wp(Q)^A$, ordered componentwise, into Q.

Given a bts $\mathcal{S} =< V, \gamma, \lambda >$ over A and L and a deterministic automaton $\mathcal{A} = < A, L, Q, q_0, \delta >$, the set $\mathbf{A}$ of *assignments* is the ordered set Q^V, the minimal element of which is $\{q_0\}^V$. The mapping δ is extended into a monotonic mapping $\delta_\mathcal{S}$ from $\mathbf{A}$ into $\mathbf{A}$ defined as in the previous case, where the notion of environment is modified to take into account the fact that the bilabelled transition system $\mathcal{S}$ is no longer deterministic:

Let α be an element of $\mathbf{A} = Q^V$; let v be an element of V; its *environment under* α is the tuple $\mathrm{env}_\mathcal{S}^\alpha(v) =< \lambda(v), \alpha(\gamma(a_1, v)), \ldots, \alpha(\gamma(a_n, v)) >$, which is an element of $L \times \wp(Q)^A$ and $\delta(\mathrm{env}_\mathcal{S}^\alpha(v))$ is an element of Q. Then $\delta_\mathcal{S}(\alpha)$ is the element of $\mathbf{A}$ defined by $\delta_\mathcal{S}(\alpha)(v) = \delta(\mathrm{env}_\mathcal{S}^\alpha(v))$.

Obviously the mapping $\delta_\mathcal{S}$ is monotononic, and since $\mathbf{A}$ has a minimal element, $\delta_\mathcal{S}$ has a least fixed point in $\mathbf{A}$ which will be denoted by $\mu\delta_\mathcal{S}$.

3.4 Homomorphisms of bts

Let $\mathcal{S}_1 =< V_1, \gamma_1, \lambda_1 >$ and $\mathcal{S}_2 =< V_2, \gamma_2, \lambda_2 >$ be two bts. An *homomorphism* from $\mathcal{S}_1$ into $\mathcal{S}_2$ is a mapping $h : V_1 \longrightarrow V_2$ such that the sets $h(\gamma_1(a, v))$ and $\gamma_2(a, h(v))$ are equal and $\lambda_1(v) = \lambda_2(h(v))$.

This definition of an homomorphism is related to the notion of bisimulation of transition systems [1]: two transition systems (all vertices of which having the same label) are in the bisimulation relation if and only if they have a common image under two surjectives homomorphisms.

It is clear from the definitions of homomorphisms and of environments that the Lemma 2.1 remains true (its proof does not change) and also the Lemma 2.2 and the Theorem 2.1.

Theorem 3.1 *If h is an homomorphism from $\mathcal{S}_1$ into $\mathcal{S}_2$, then*

$$\mu\delta_{\mathcal{S}_1} = \mu\delta_{\mathcal{S}_2} \circ h$$

3.5 Other extensions

The value of the transition function δ of an automaton at some vertex v of a transition system $\mathcal{S}$ under an assignment α depends on the environment of v under α, which consists in the label of v and the image under α of some sets of vertices associated with v. One can imagine others transition functions which depend on larger environments of a vertex. Here we present two such extensions. In the first one δ take into account the state assigned to the vertex and we show that this extension is not more powerful. In the second one we put in the environment of v under α the sets $\gamma^{-1}(a, v)$, equal to $\{v' | v \in \gamma(a, v)\}$. If one considers bts automata as bottom-up automata because states are propagated back along the edges (the state of v depends on the states of $\gamma(a, v)$), then top-down automata are those where only the sets $\gamma^{-1}(a, v)$ are in the environments, instead of the sets $\gamma(a, v)$, and, when both are in the environment, we get something like a bidirectional automaton.

3.5.1 Vertices need not to be in their environment.

Here we assume that the transition function of an automaton is a monotonic mapping from $Q \times L \times \wp(Q)^A$, ordered componentwise, into Q. Then $\delta_S(\alpha)$ is the element of $\mathbf{A}$ defined by $\delta_S(\alpha)(v) = \delta(\alpha(v), \mathrm{env}_S^\alpha(v))$.

Let us define the mapping δ^* from $L \times \wp(Q)^A$ into Q by:
$\delta^*(l, X_{a_1}, \ldots, X_{a_n})$ is the limit of the sequence $(q_i)_{i \leq 0}$ defined by

- q_0 is the least element of Q,
- $q_{i+1} = \delta(q_i, l, X_{a_1}, \ldots, X_{a_n})$.

This sequence is increasing and we have

$$\delta^*(l, X_{a_1}, \ldots, X_{a_n}) = \delta(\delta^*(l, X_{a_1}, \ldots, X_{a_n}), l, X_{a_1}, \ldots, X_{a_n})$$

Let α and β be the least fixed points of δ_S and of δ_S^*. Then we can show that $\alpha = \beta$.

From the definitions of δ_S^*, δ_S, and by the previous equality, we have

$$\begin{aligned}\beta(v) &= \delta_S^*(\beta)(v)\\ &= \delta^*(\mathrm{env}_S^\beta(v))\\ &= \delta(\delta^*(\mathrm{env}_S^\beta(v)), \mathrm{env}_S^\beta(v))\\ &= \delta(\beta(v), \mathrm{env}_S^\beta(v))\\ &= \delta_S(\beta)(v)\end{aligned}$$

and then α is less than β.

Conversely, let us define β as the limit of the increasing sequence $(\beta_i)_{i \leq 0}$ with $\beta_0 = \perp$ and $\beta_{i+1} = \delta_S^*(\beta_i)$. We prove by induction that β_i is less than α.

Obviously, β_0 is less than α.

Since $\beta_{i+1} = \delta_S^*(\beta_i)$, since, by induction hypothesis, β_i is less than α, and since δ_S^* is monotonic, we get

$$\beta_{i+1} \leq \delta_S^*(\alpha)$$

and it remains to prove:

$$\delta_S^*(\alpha) \leq \alpha$$

For any v we have

$$\delta_S^*(\alpha)(v) = \delta^*(\mathrm{env}_S^\alpha(v))$$

and let $\delta^*(\mathrm{env}_S^\alpha(v))$ be the limit of the increasing sequence $(q_i)_{i \leq 0}$ with $q_0 = \perp$ and $q_{i+1} = \delta(q_i, \mathrm{env}_S^\alpha(v))$. We prove by induction that $q_i \leq \alpha(v)$, hence the result.

q_0 is less than $\alpha(v)$. Let us assume that q_i is less than α. Then $q_{i+1} = \delta(q_i, \mathrm{env}_S^\alpha(v))$, and since δ is monotonic, $q_{i+1} \leq \delta(\alpha(v), \mathrm{env}_S^\alpha(v))$. But
$\delta(\alpha(v), \mathrm{env}_S^\alpha(v)) = \delta_S(\alpha)(v) = \alpha(v)$, which ends the proof.

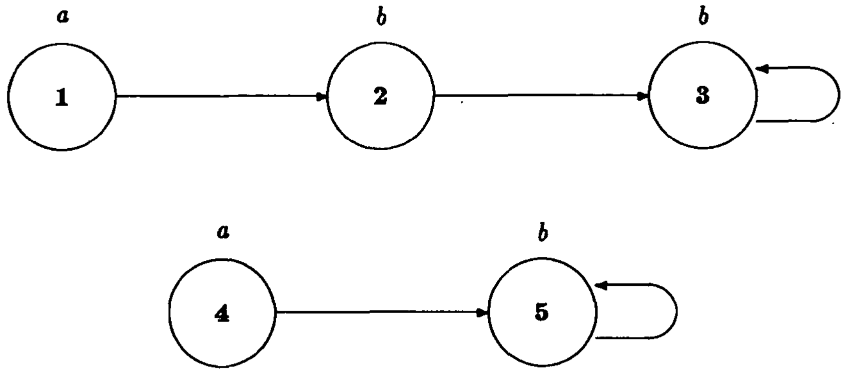

Figure 1.

3.5.2 Bidirectional automata

Let us consider the two following cdbts pictured in figure 1 with $\{a, b\}$ as set of labels and a single-letter alphabet.

Let us consider the set $Q = \{q_0, q_1, q_2\}$ with $q_0 \leq q_1 \leq q_2$, and the function δ from $\{a, b\} \times \wp(Q) \times \wp(Q)$ int› Q, where the second argument of δ corresponds to γ^{-1} and the third one to γ, dcfined by:

$$\begin{aligned}\delta(a, X, Y) &= q_0, \forall X, Y \subseteq Q \\ \delta(b, \{q_0\}, Y) &= q_1, \forall Y \subseteq Q \\ \delta(b, X, Y) &= q_2, \forall X, Y \subseteq Q, X \neq \{q_0\}\end{aligned}$$

Then, on the first bts of the figure 1, the assigment α which is the least fixed point of δ is

$$\begin{aligned}\alpha(\mathbf{1}) &= q_0 \\ \alpha(\mathbf{2}) &= q_1 \\ \alpha(\mathbf{3}) &= q_2\end{aligned}$$

and for the second one, the least assignment β is

$$\begin{aligned}\beta(\mathbf{4}) &= q_0 \\ \beta(\mathbf{5}) &= q_2\end{aligned}$$

On the other hand, the mapping which sends **1** on **4** and **2** and **3** on **5** is a surjective bts homomorphism, and because of Theorem 2.1, the least assigment α and β of a "bottom-up"deterministic bts automaton should satisfy $\alpha(\mathbf{1}) = \beta(\mathbf{4})$ and $\alpha(\mathbf{2}) = \alpha(\mathbf{3}) = \beta(\mathbf{5})$, which shows up that bidirectional bts automata are more powerful than bottom-up bts automata.

4 Products of deterministic bts automata

Let $\mathcal{A}_1 =< A, L, Q_1, i_1, \delta_1 >$ and $\mathcal{A}_2 =< A, L, Q_2, i_2, \delta_2 >$, where

$$\delta_i : L \times \wp(Q_i)^A \longrightarrow Q_i$$

for $i = 1, 2$, be two deterministic bts automata over the alphabet A and the labels L.

The product $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ of $\mathcal{A}_1$ and $\mathcal{A}_2$ is the deterministic bts automaton $< A, L, Q, i, \delta >$ with

$Q = Q_1 \times Q_2$,

$i =< i_1, i_2 >$,

$\delta : L \times \wp(Q_1 \times Q_2)^A \longrightarrow Q_1 \times Q_2$ defined by

$\delta(l, P_{a_1}, \ldots, P_{a_n}) =< \delta_1(l, \pi_1(P_{a_1}), \ldots, \pi_1(P_{a_n})), \delta_2(l, \pi_2(P_{a_1}), \ldots, \pi_2(P_{a_n})) >$
where π_i is the canonical projection of $Q_1 \times Q_2$ onto Q_i.

Indeed, it is straightforward from its definition that δ is monotonic because it is very easy to prove that if $P \sqsubseteq P'$, $P, P' \subseteq Q_1 \times Q_2$, then $\pi_i(P) \sqsubseteq \pi_i(P')$.

Let us consider a bts $S =< V, \gamma, \lambda >$. If α is an assigment in $(Q_1 \times Q_2)^V$, it can be seen as the product $\alpha_1 \times \alpha_2$ of the assignments α_i in Q_i^V defined by $\alpha_1 \times \alpha_2(v) = < \alpha_1(v), \alpha_2(v) >$. It follows:

$$\delta_S(\alpha) = \delta_{1S}(\alpha_1) \times \delta_{2S}(\alpha_2)$$

and the least fixed point $\mu\delta_S$ of δ is equal to the product $\mu\delta_{1S} \times \mu\delta_{2S}$ of the least fixed points of δ_{1S} and δ_{2S}.

Theorem 4.1 *If $\delta = \delta_1 \times \delta_2$ then $\mu\delta_S = \mu\delta_{1S} \times \mu\delta_{2S}$*

5 Acceptance criterion

In order to have a notion of recognizable set of bts we have to define an acceptance criterion for the bts automata. The criterion we choose is such that boolean combinations of recognizable sets are still recognizable, which is a very natural assumption.

5.1 Definitions

Let $\mathcal{A} =< A, L, Q, i, \delta >$ be a bts automaton. An acceptance criterion for $\mathcal{A}$ is a set $\mathcal{F}$ of subsets of Q.

Given a bts $S =< V, \gamma, \lambda >$ and the least fixed point $\mu\delta_S$ in Q^V of δ, we say that S is accepted by the pair $< \mathcal{A}, \mathcal{F} >$ iff $\mu\delta_S(V)$ is an element of $\mathcal{F}$. A recognizable set is the set of all bts accepted by some pair $< \mathcal{A}, \mathcal{F} >$.

5.2 Properties of recognizable sets

Let $B(A, L)$ the set of all bts over the alphabet A and the labels L.

The following property is a straightforward consequence of the definitions:

Lemma 5.1 *If R is a subset of $B(A, L)$ recognized by the pair $< \mathcal{A}, \mathcal{F} >$ then its complement is recognized by the pair $< \mathcal{A}, \wp(Q) - \mathcal{F} >$.*

Let us consider two bts automata $\mathcal{A}_j =< A, L, Q_j, i_j, \delta_j >$ for $j = 1, 2$, and two acceptance criteria $\mathcal{F}_j$ in $\wp(Q_j)$. Let us denote by $\mathcal{G}_\cup$ (resp. $\mathcal{G}_\cap$) the set of all subsets F of $Q_1 \times Q_2$ such that $\pi_1(F) \in \mathcal{F}_1$ or (resp. and) $\pi_2(F) \in \mathcal{F}_2$ where π_j is the canonical projection of $Q_1 \times Q_2$ on Q_j.

Lemma 5.2 *A bts is accepted by the pair $< \mathcal{A}_1 \times \mathcal{A}_2, \mathcal{G}_\cup >$ iff it is accepted by the pair $< \mathcal{A}_1, \mathcal{F}_1 >$ or by the pair $< \mathcal{A}_2, \mathcal{F}_2 >$.*
A bts is accepted by the pair $< \mathcal{A}_1 \times \mathcal{A}_2, \mathcal{G}_\cap >$ iff it is accepted by the pair $< \mathcal{A}_1, \mathcal{F}_1 >$ and by the pair $< \mathcal{A}_2, \mathcal{F}_2 >$.

Proof We consider only the first case; the second one is proved exactly the same way.
From the definition of the product of automata we have (Theorem 4.1)

$$\mu\delta_S = \mu\delta_{1S} \times \mu\delta_{2S}$$

Hence $\pi_i(\mu\delta_S(V)) = \mu\delta_{iS}(V)$, and $\mu\delta_S(V)$ belongs to $\mathcal{G}_\cup$ iff $\mu\delta_{1S}(V)$ belongs to $\mathcal{F}_1$ or $\mu\delta_{2S}(V)$ belongs to $\mathcal{F}_2$. □

Theorem 5.1 *The set of all recognizable subsets of $B(A, L)$ is a boolean algebra.*

Proof By lemma 5.1, this set is closed under complement. By lemma 5.2 the union and the intersection of the two subsets recognized by the pairs $< \mathcal{A}_1, \mathcal{F}_1 >$ and $< \mathcal{A}_2, \mathcal{F}_2 >$ are recognized by the pairs $< \mathcal{A}_1 \times \mathcal{A}_2, \mathcal{G}_\cup >$ and $< \mathcal{A}_1 \times \mathcal{A}_2, \mathcal{G}_\cap >$. □

A less usual property is:

Proposition 5.1 *Let $h : S_1 \longrightarrow S_2$ be a* surjective *homomorphism of bts. Then S_1 is accepted by the pair $< \mathcal{A}, \mathcal{F} >$ iff S_2 is accepted by this pair.*

Proof We already know, by Theorem 3.1 that $\mu\delta_{S_1} = \mu\delta_{S_2} \circ h$. Hence $\mu\delta_{S_1}(V) = \mu\delta_{S_2}(h(V)) = \mu\delta_{S_2}(V') = Q$. □

6 Fibers

Let us define the following binary relation between bts: two bts are in the relation if they have a common homomorphic image. It can be proved (see [1]) that this is an equivalence relation and, moreover, that each equivalence class has a canonical representant which is minimal in the following sense: it is an homomorphic image of every bts in its equivalence class. Therefore we call *fiber* such an equivalence class. From proposition 5.1 every recognizable set is a union of fibers. Here we prove that **every fiber is recognizable.**

Let $S =< V, \gamma, \lambda >$ be a bts. Let us define, for every vertex v and for every natural number n the bts $T(v, n)$, which is indeed a tree, as follows:

- $T(v, 0)$ is a single vertex labelled by a special symbol, say $\perp$,

- $T(v,n+1)$ is the disjoint union of all $T(v',n)$ for all v' in $\bigcup_{a\in A}\gamma(a,v)$ together with a new vertex v_0 labelled by $\lambda(v)$ and, for any letter a in A, a set of edges labelled by a from v_0 to every $T(v',n)$, for v' in $\gamma(a,v)$.

For short, $T(v,n+1)$ can be written as $\lambda(v)(\bigcup_{a\in A}\bigcup_{v'\in\gamma(a,v)} a\cdot T(v',n))$.

The set of all such $T(v,n)$ can be ordered, by induction on n, by:

- $T(v,0)$ is less than anything,
- $T(v,n+1)$ is less than $T(v',n'+1)$, for $n\leq n'$ iff
  - $\lambda(v)=\lambda(v')$,
  - for any a in A, the set $\{T(v'',n)\mid v''\in\gamma(a,v)\}$ is less (with respect to the induced Egli–Milner ordering) than the set $\{T(v'',n')\mid v''\in\gamma(a,v')\}$.

Lemma 6.1 *If* $T(v,n+1)=T(v',n+1)$, *then* $T(v,n)=T(v',n)$.

Proof The proof is by induction on n.
•If $T(v,1)=t(v',1)$, then obviously $T(v,0)=T(v',0)=\perp$.

•If $T(v,n+2)=T(v',n+2)$, then $\lambda(v)=\lambda(v')$ and for any a, $\{T(v'',n+1)\mid v''\in\gamma(a,v)\}=\{T(v'',n+1)\mid v''\in\gamma(a,v')\}$. Then for any $u\in\gamma(a,v)$, (resp. $\in\gamma(a,v')$) there exists $u'\in\gamma(a,v')$ (resp. $\in\gamma(a,v)$) such that $T(u,n+1)=T(u',n+1)$. By induction hypothesis, $T(u,n)=T(u',n)$; hence $\{T(v'',n)\mid v''\in\gamma(a,v)\}=\{T(v'',n)\mid v''\in\gamma(a,v')\}$ and $T(v,n+1)=T(v',n+1)$. □

Then we define the following family of equivalence relations between the vertices of a bts.

$$v\overset{i}{\sim}v' \text{ iff } T(v,i)=T(v',i)$$

By the previous lemma we get

$$v\overset{i+1}{\sim}v'\Rightarrow v\overset{i}{\sim}v'$$

Let k_i be the number of classes of $\overset{i}{\sim}$. We have

- $1\leq k_i\leq|V|$, where $|V|$ is the number of elements of V;
- $k_i\leq k_{i+1}$;
- if $k_i=k_{i+1}$ then $\overset{i}{\sim}=\overset{i+1}{\sim}=\overset{i+j}{\sim}$ for every $j\geq 1$.

This last point can be easily proved in the following way: since $\overset{i+1}{\sim}$ is included in $\overset{i}{\sim}$, if they have the same number of classes, they are equal. Now let us assume that $\overset{n}{\sim}=\overset{n+1}{\sim}$. $T(v,n+2)$ can be written as

$$\lambda(v)(\bigcup_{a\in A}\bigcup_{v'\in\gamma(a,v)} a\cdot T(v',n+1)),$$

and $T(u, n+2)$ as

$$\lambda(u)(\bigcup_{a\in A} \bigcup_{v'\in\gamma(a,u)} a \cdot T(v', n+1)).$$

If they are equal, it means that $\lambda(v) = \lambda(u)$ and that for every a, if $v' \in \gamma(a, v)$, there is some $u' \in \gamma(a, u)$ (and conversely) such that $T(v', n+1) = T(u', n+1)$, hence $T(v', n) = T(u', n)$ and $T(v, n+1) = T(u, n+1)$.

Thus we get the following lemma:

Lemma 6.2 $\overset{|V|}{\sim} = \overset{|V|+1}{\sim}$

Let us define $\sim$ to be $\overset{|V|}{\sim}$ and let us denote by $[v]$ the equivalence class of v for $\sim$, and for any subset V' of V, by $[V']$ the set $\{[v] \mid v \in V'\}$. We can define the quotient bts $S/\sim = < [V], \gamma', \lambda' >$ of S in the following way:

- $\lambda'([v]) = \lambda(v)$; this is independent on the choice of v in $[v]$ because of the very definition of $\sim$;
- $\gamma'(a, [v]) = [\gamma(a, v)]$; this is also independent on the choice of v because $v \sim v'$ implies $v \overset{|V|+1}{\sim} v'$, hence $T(v, |V|+1) = T(v', |V|+1)$; it follows that $\{T(v'', |V|) \mid v'' \in \gamma(a, v)\} = \{T(v'', |V|) \mid v'' \in \gamma(a, v')\}$, hence $[\gamma(a, v)] = [\gamma(a, v')]$.

It follows from this definition that $S/\sim$ is the image of S under a surjective homomorphism, from which we derive the lemma:

Lemma 6.3 *If S is minimal, then $S = S/\sim$, and this implies that $v \neq v' \Rightarrow T(v, |V|) \neq T(v', |V|)$.*

Now let us consider a minimal bts S. Let us define the finite set Q as beeing the set of all (tree-shaped) transition systems obtained by deleting some subtrees in some $T(v, |V|)$ and replacing them by a vertex labelled by $\bot$. This set is obviously ordered and contains $\{T(v, n) \mid v \in V, n \leq |V|\}$ as an ordered subset. Let us define the following bts automaton $\mathcal{A} = < A, L, Q \cup \{\sigma\}, \bot, \delta >$ where Q is defined as above, σ is a new state greater than any other state and where δ is defined by

$$\delta(l, X_{a_1}, \ldots, X_{a_n}) = \tau$$

where τ is defined as follows:
•first, let τ' be

$$l(\bigcup_{a\in A} \bigcup_{v\in X_a} a \cdot v),$$

•second, replace all subtrees of τ' at depth greater than $|V|$ by $\bot$, getting τ'',
•then if $\tau'' \in Q$ then $\tau = \tau''$ otherwise $\tau = \sigma$.

It is easy to see that δ is increasing.

Finally let us define the acceptance condition as consisting only of the set $F = \{T(v, |V|) \mid v \in V\}$ which is obviously included in Q.

Let now $\mu\delta_S$ the least fixed point of the transition function δ over S. Then it can be shown that $\mu\delta_S(v) = T(v, |V|)$: clearly all the states assigned to v during the computation of $\mu\delta_S$ are less than $T(v, |V|)$; let us assume that some $\mu\delta_S(v)$ is strictly less than $T(v, |V|)$; that means that in this state $\mu\delta_S(v)$, $\perp_\emptyset$ appears at a depth less than $|V|$ which implies by an inductive argument that $\mu\delta_S(v') = \perp$ for some v' which is impossible because every vertex is assigned at least a tree whose root is labelled by the label of this vertex.

Thus S is accepted by the pair $< \mathcal{A}, \{F\} >$.

Conversely, let us assume that some S' is also accepted by the pair $< \mathcal{A}, \{F\} >$. We have to show that S is a homomorphic image of S'.

Let $\alpha = \mu\delta'_S$. Let us define the mapping associating with a vertex v of S' a vertex u of S such that $\alpha(v) = T(u, |V|)$; the result does not depend on the choice of u since, S beeing minimal, by lemma 6.3, $T(u, |V|) = T(u', |V|)$ implies $u = u'$. Moreover it is surjective since for every vertex v of S, there exists a vertex v' of S' such that $\alpha(v') = T(v, |V|)$. It remains to prove that this mapping is a bts homomorphism, which is a straightforward consequence of the fact that α is a fixed point of the transition function δ and of the definition of δ.

7 Non deterministic bts automata

The bts automata previously defined are deterministic in the sense that the transition relation δ is indeed a function from $L \times \wp(Q)^A$ into Q. If an automaton is non deterministic, then δ has to be one-to-many, but in this case it is difficult to define a condition of monotonicity which guarantees the existence of a least fixed point.

A first approach to this problem is to consider that a non deterministic transition relation is not a one-to-many mapping but a set of functions: applying such a transition relation consists in choosing one of the function in the set and applying it. In this case, one has just to assume that each one of these functions is monotonic. Such a point of view about non deterministic functions (sets of deterministic functions rather than multivocal functions) has already been fruitfully applied to the semantics of non deterministic recursive program schemes [2]. For the classical cases (words, trees), a transition function can always be defined this way provided two sets $\mathcal{F}$ and $\mathcal{F}'$ of functions are considered equivalent if for every argument x the two sets $\{f(x) \mid f \in \mathcal{F}\}$ and $\{f(x) \mid f \in \mathcal{F}'\}$ are equal, which means that the two sets define the same multivocal function.

This is probably not enough to guarantee the exixtence of a least fixed point: each time a vertex is "visited" one has to apply one of the transition functions appliable to this vertex; since it is not necessarily the same one which is chosen at every visit, there is no reason for the value of the state assigned to this vertex only increases. This problem disappears if, once one of the function appliable to some vertex is chosen, at each further visit, this function will be chosen too. In this case we are sure that a least fixed point will be reached.

But then, it is equivalent to add to each vertex another label, which indicates which is the function to be applied at this vertex and, on such a transition system, the automaton becomes deterministic. Therefore one can say that a set of bts is *non deterministically recognizable* if it is the *projection* of a set recognized by a (deterministic) bts automaton, where the projection of a bts is defined as follows:

A bts $S =< V, \gamma, \lambda >$ over A and L is a *projection* of $S' =< V', \gamma', \lambda' >$ over A and L' if there exist a bijection β between V' and V and a mapping p from L' in L such that

- for every v in V, $\lambda'(\beta(v)) = p(\lambda(v))$;

- for every v in V and for every a in A, $\gamma'(a, \beta(v)) = \beta(\gamma(a, v))$.

In other words, S is obtained from S' by replacing the label of every vertex by its image under p.

An example Let us define the binary operator $\oplus$ which associates, with two bts, their disjoint union. Let us extend this operator to sets of bts by

$$K \oplus K' = \{S \oplus S' \mid S \in K, S' \in K'\}.$$

It is easy to see that if K and K' are both recognizable by (deterministic) automata, then $K \oplus K'$ is non deterministically recognizable:

- first of all, consider two isomorphic copies K_1 and K_2 of K and K', with disjoints sets of vertex labels.
- K_1 and K_2 are still recognizable by two deterministic automata and one can assume that they have disjoint sets of states.
- the "disjoint union" (the intuitive meaning of this notion is obvious) is still a deterministic bts automaton and one can easily define an acceptance criterion such that it recognizes $K_1 \oplus K_2$.

Then $K \oplus K'$ is the projection of $K_1 \oplus K_2$. Intuitively speaking the modifications of the vertex labels of K and K' simply allows to select which automaton has to run when visiting some vertex of K or K'.

On the other hand the disjoint union (in the sense defined above) of two deterministically recognizable sets is not necessarily deterministically recognizable. Indeed, it seems probable that there exist examples of deterministically recognizable sets, disjoint union of which is not. But it remains to find out such examples and to show that their disjoint union is not deterministically recognizable, which could be not so easy.

Also the family of nondeterministically recognizable sets need not to be a boolean algebra: it is obviously closed under union but probably not under complementation, nor even under intersection. Here again counter-examples and proofs have to be given and are presumably not immediate.

8 Recognizable sets and graph grammars

It is well known that in the cases of words and of trees, there exist some kinds of grammars, the regular grammars, which generate exactly the recognizable sets of words and of trees. The question is quite open for the case studied here: a lot of different kinds of graph grammars have already been defined in the litterature; do there exist some kinds of graph grammars which generate exactly the deterministically and/or the nondeterministically recognizable sets of bilabelled transition systems, and which, therefore, will be deserved to be named regular, as far as the notion of recognizability defined in this paper can be considered as a correct extension of the similar notion for words and trees.

References

[1] A. Arnold and A. Dicky. *An algebraic characterization of transition system equivalences.* Technical Report I-8603, Université de Bordeaux I, 1986.

[2] A. Arnold, P. Naudin, and M. Nivat. On semantics of non deterministic program schemes. In M. Nivat and J. Reynolds, editors, *Algebraic methods in semantics*, pages 1–33, Cambridge University Press, 1985.

[3] F. Bossut and B.Warin. *Rationalité et reconnaissabilité dans des graphes.* PhD thesis, Université de Lille, 1986.

[4] J. R. Büchi. On a decision method in restricted second order arithmetic. In E. Nagl, editor, *Logic, Methodology, and Philosophy of Science*, pages 1–11, Stanford Univ. Press, 1960.

[5] B. Courcelle. Some applications of logic of universal algebra, and of category theory to the theory of graph transformations. *Bull. EATCS*, 36:161–218, october 1988.

[6] D. Kozen. Results on the propositional μ-calculus. *Theoretical Comput. Sci.*, 27:333–354, 1983.

[7] G. D. Plotkin. A powerdomain construction. *SIAM J. Comput.*, 5:452–487, 1976.

[8] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

(Received September 25, 1989)

Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1990

A note on axiomatizing flowchart schemes

V. E. Cazanescu [*] Gh. Ştefanescu[†]

Abstract

A biflow is an equationally presented algebraic structure which completely characterizes flowchart schemes from the algebraic point of view. Usually it is presented using summation, composition, (left or right) feedbackation, identities, and block transpositions. In the present paper we give a new presentation for the biflow structure, without making use of composition and block transpositions, but using an extended feedbackation.

1 Introduction

The axiomatization of flowchart schemes is a basic step toward an algebraic theory of computation (see [CS87b], for example). A series of papers [CU82 & CG84], [BE85], [St86b], [Ba87] and [CS87a & 88b] has lead to an algebraic structure, called biflow, which completely characterizes flowchart schemes from the algebraic point of view. This structure uses a new looping operation, called feedbackation (introduced in [St86a]), which is in some cases better than iteration or repetition, cf. [St86b], [CS88a].

The biflow structure has been introduced in [St86b], without axiomatizing bijections. The bijections were axiomatized in [Ba87] and [CS87a] leading to the actual presentation for the biflow structure.

A usual flowchart scheme is built up from some atomic schemes connected by arrows. Note that a natural structure of biflow may be given on the collection of the sets of arrows used to connect atomic elements in flowchart schemes. The main interest in the biflow structure comes from the following theorem [CS88b, Theorem 8.2] (also presented in Section 4 of [CS87b]):

> *"Flowchart schemes have a universal property similar to that of polynomials, i.e. the flowchart schemes obtained using elements in a biflow T, as connections, and elements in a double-ranked set X, as atomic schemes, form the biflow freely generated by adding X to T."*

Hence in our theory of program schemes the biflow of flowchart schemes have the same role as the ring of polynomials in classical algebra. In other words the axioms which define the biflow structure give a complete axiomatization for the concrete

[*]Faculty of Mathematics, University of Bucharest, Str. Academiei 14, 70109 Bucharest, ROMANIA

[†]Institute of Mathematics, Romanian Academy of Sciences, Bd. Pacii 220, 79622 Bucharest, ROMANIA

Table 1: Axioms B1-16 define a biflow, while B1-5, B6a,b,c, B7a, B8-16 define a flow.

$B1$	$f \cdot (g \cdot h) = (f \cdot g) \cdot h$		
$B2$	$\mathrm{I}_a \cdot f = f = f \cdot \mathrm{I}_b$		
$B3$	$f + (g + h) = (f + g) + h$		
$B4$	$\mathrm{I}_0 + f = f = f + \mathrm{I}_0$		
$B5$	$\mathrm{I}_a + \mathrm{I}_b = \mathrm{I}_{ab}$		
$B6$	$(f + f') \cdot (g + g') = f \cdot g + f' \cdot g'$	$B6a$	$(f + \mathrm{I}_{b'}) \cdot (g + g') = f \cdot g + \mathrm{I}_{b'} \cdot g'$
		$B6b$	$(f + f') \cdot (\mathrm{I}_b + g') = f \cdot \mathrm{I}_b + f' \cdot g'$
		$B6c$	$(\mathrm{I}_{ab}+f) \cdot ({}^a\mathrm{X}^b+\mathrm{I}_d) = \mathrm{I}_{ab} \cdot {}^a\mathrm{X}^b+f \cdot \mathrm{I}_d$
$B7$	${}^c\mathrm{X}^a \cdot (f + g) \cdot {}^b\mathrm{X}^d = g + f$ for $f : a \rightarrow b, g : c \rightarrow d$	$B7a$	${}^c\mathrm{X}^a \cdot (f + \mathrm{I}_c) \cdot {}^b\mathrm{X}^c = \mathrm{I}_c + f$ for $f : a \rightarrow b$
$B8$	${}^a\mathrm{X}^0 = \mathrm{I}_a$		
$B9$	${}^a\mathrm{X}^{bc} = ({}^a\mathrm{X}^b + \mathrm{I}_c) \cdot (\mathrm{I}_b +{}^a \mathrm{X}^c)$		
$B10$	$f \cdot g \uparrow^a = ((f + \mathrm{I}_a) \cdot g) \uparrow^a$		
$B11$	$f \uparrow^a \cdot g = (f \cdot (g + \mathrm{I}_a)) \uparrow^a$		
$B12$	$f + g \uparrow^a = (f + g) \uparrow^a$		
$B13$	$(f \cdot (\mathrm{I}_d +{}^a \mathrm{X}^b)) \uparrow^{ba} = ((\mathrm{I}_c +{}^a \mathrm{X}^b) \cdot f) \uparrow^{ab}$		
$B14$	$f \uparrow^{ab} = f \uparrow^b \uparrow^a$		
$B15$	$\mathrm{I}_a \uparrow^a = \mathrm{I}_0$		
$B16$	${}^a\mathrm{X}^a \uparrow^a = \mathrm{I}_a$		

flowchart schemes in the context of Manysorted Equational Logic. An element in an abstract biflow may be regarded as an abstract flowchart scheme.

A comparison between feedbackation, Elgot's iteration and Kleene's repetition is given in [CS88a]. As a by-product there are given certain axiom systems for defining the concept of "biflow over an algebraic theory", in terms of iteration, and the concept of "biflow over a matrix theory", in terms of repetition.

The aim of this paper is to give an axiom system for defining the biflow structure, without making use of composition and block transpositions, but using an extended feedbackation, i.e. we allow a feedback to connect an arbitrary output with an arbitrary, compatible input. (In the previous papers we have used only the right and the left feedbackation.) Consequently the operations we use in this paper are: *summation,* (extended) *feedbackation*, and *identities*.

The inspiration for the present note came from a reading of Parrow's axiomatization [Pa87] for a certain kind of nets, called synchronization flow networks. A comparison between the present axiom system for biflow and Parrow's axiom system is given [CS89].

* * *

Let $(M, +, 0)$ be a free monoid. We agree to omit the writing of "+", that is we write, for example, ab insted of $a + b$. The letters a, b, c, d, e, u, v, w will denote elements of M.

Definition 1 *An* M-biflow *(resp.* M-flow*) B is an abstract structure given by:*

- *a family of sets* $\{B(a, b)\}_{a,b \in M}$;

Table 2: These axioms define a BIFLOW, while the subset of all the axioms denoted by S.. or F.. define a FLOW

$S1$	$f+(g+h)=(f+g)+h$		
$S2$	$f+\mathrm{I}_0=f$	$S2^\circ$	$f=\mathrm{I}_0+f$
$S3$	$\mathrm{I}_a+\mathrm{I}_b=\mathrm{I}_{ab}$		
$F0a$	$f\uparrow^{a(0)b}_{c(0)d}=f$		
$F0b$	$f\uparrow^{a(uv)b}_{c(uv)d}=f\uparrow^{a(u)vb}_{c(u)vd}\uparrow^{a(v)b}_{c(v)d}$	$F0b^\circ$	$f\uparrow^{bv(u)a}_{dv(u)c}\uparrow^{b(v)a}_{d(v)c}=f\uparrow^{b(vu)a}_{d(vu)c}$
$F1$	$f\uparrow^{a(u)b}_{c(u)d}+g=(f+g)\uparrow^{a(u)ba'}_{c(u)dc'}$	$F1^\circ$	$(g+f)\uparrow^{a'b(u)a}_{c'd(u)c}=g+f\uparrow^{b(u)a}_{d(u)c}$
$F2a$	$f\uparrow^{a(u)bvc}_{a'(u)b'vc'}\uparrow^{ab(v)c}_{a'b'(v)c'}=f\uparrow^{aub(v)c}_{a'ub'(v)c'}\uparrow^{a(u)bc}_{a'(v)b'c'}$		
$F2b$	$f\uparrow^{a(u)bvc}_{a'vb'(u)c'}\uparrow^{ab(v)c}_{a'(v)b'c'}=f\uparrow^{aub(v)c}_{a'(v)b'uc'}\uparrow^{a(u)bc}_{a'b'(u)c'}$		
$F3$	$(\mathrm{I}_a+f)\uparrow^{a(a)0}_{0(a)b}=f$	$F3^\circ$	$f=(f+\mathrm{I}_a)\uparrow^{0(a)a}_{b(a)0}$
$F3_\circ$	$(\mathrm{I}_a+f)\uparrow^{0(a)b}_{a(a)0}=f$	$F3^\circ_\circ$	$f=(f+\mathrm{I}_a)\uparrow^{b(a)0}_{0(a)a}$
$F4$	$\mathrm{I}_a\uparrow^{0(a)0}_{0(a)0}=\mathrm{I}_0$		
P	$(f+g)\uparrow^{a(b)0}_{0(b)c}=(g+f)\uparrow^{0(b)a}_{c(b)0}$		

- *two kinds of constants:*
  Identities $\mathrm{I}_a\in B(a,a)$ for $a\in M$ and
  Block Transpositions ${}^a\mathrm{X}^b\in B(ab,ba)$ for $a,b\in M$;
- *three operations:*
  Summation $+$:$B(a,b)\times B(c,d)\to B(ac,bd)$ for $a,b,c,d\in M$,
  Composition $\cdot$: $B(a,b)\times B(b,c)\to B(a,c)$ for $a,b,c\in M$, and
  (Right) Feedbackation $\uparrow^a$: $B(ba,ca)\to B(b,c)$ for $a,b,c\in M$

and fulfilling axioms B1-16 (*resp.* B1-5, B6a,b,c, B7a *and* B8-16) *in Table 1*[1]. □

We declare that feedbackation has the strongest binding power, then composition, then summation. For instance, $f\cdot g+f'\cdot g'$ means $(f\cdot g)+(f'\cdot g')$, while $f+g\uparrow^a$ means $f+(g\uparrow^a)$. The sign of composition " $\cdot$ " is usually omitted.

In an M-(bi)flow B, defined as above, one may define an extended feedbackation $\Uparrow^{a(u)b}_{c(u)d}$ by

$$f\Uparrow^{a(u)b}_{c(u)d}=[(\mathrm{I}_a+{}^b\mathrm{X}^u)\cdot f\cdot(\mathrm{I}_c+{}^u\mathrm{X}^d)]\uparrow^u \text{ for } f\in B(aub,cud). \tag{1}$$

Definition 2 *Let us say that B is an* M-BIFLOW (*resp.* M-FLOW) *if it is given by a family of sets $\{B(a,b)\}_{a,b\in M}$, a summation, identities, and an*

Extended Feedbackation $\uparrow^{a(u)b}_{c(u)d}$: $B(aub,cud)\to B(ab,cd)$ for $a,b,c,d,u\in M$

and fulfilling the axioms in Table 2 (resp. all the axioms in Table 2 denoted by strings starting with letter "S" or "F"). □

[1]The algebra $Fl_{\Sigma,Pfn}$ of representations of Σ-flowcharts over Pfn in [CS87a] is an example of flow which is not a biflow (provided $\Sigma\neq 0$).

Consequently an M-BIFLOW is an M-FLOW which fulfills axiom (P).

In an M-(BI)FLOW B one may define a composition "o" by

$$f \circ g = (f+g)\uparrow^{a(b)0}_{0(b)c} \text{ for } f \in B(a,b), g \in B(b,c), \qquad (2)$$

block transpositions ${}^a\mathrm{X}^b$ by

$$^a\mathrm{X}^b = (\mathrm{I}_a + \mathrm{I}_b + \mathrm{I}_a)\uparrow^{ab(a)0}_{0(a)ba} \text{ for } a,b \in M \qquad (3)$$

and a (right) feedbackation $\_\uparrow^a$ by

$$f\uparrow^a = f\uparrow^{b(a)0}_{c(a)0} \text{ for } f \in B(ba,ca). \qquad (4)$$

The aim of this note is to prove that the concept of M-biflow coincides with the concept of M-BIFLOW, i.e. the above definitions give two different presentations of the same algebraic structure. As a by-product we also get the equivalence of the concepts of M-flow and M-FLOW.

More precisely, let us consider the following transformations

$$\mathbf{B} = (B, +, \cdot, \uparrow^a, \mathrm{I}_a, {}^a\underline{\mathrm{X}}^b) \mapsto \alpha(\mathbf{B}) = (B, +, \Uparrow^{a(u)b}_{c(u)d}, \mathrm{I}_a),$$

where $\Uparrow^{a(u)b}_{c(u)d}$ is that defined in $\mathbf{B}$ by formula (1) and

$$\mathbf{C} = (C, +, \uparrow^{a(u)b}_{c(u)d}, \mathrm{I}_a) \mapsto \beta(\mathbf{C}) = (C, +, \circ, \Uparrow^a, \mathrm{I}_a, {}^a\mathrm{X}^b)$$

where $\circ$, ${}^a\mathrm{X}^b$, and $\Uparrow^a$ are those defined in $\mathbf{C}$ by formulas (2), (3), (4), respectively. We shall prove the following theorem.

Theorem 1 (i) *If* $\mathbf{B}$ *is an M-biflow, then* $\alpha(\mathbf{B})$ *is an M-BIFLOW.*

(ii) *If* $\mathbf{C}$ *is an M-BIFLOW, then* $\beta(\mathbf{C})$ *is an M-biflow.*

(iii) *If* $\mathbf{B}$ *is an M-biflow then* $\beta(\alpha(\mathbf{B})) = \mathbf{B}$, *and if* $\mathbf{C}$ *is an M-BIFLOW then* $\alpha(\beta(\mathbf{C})) = \mathbf{C}$.

The proof of this theorem is based on the analogous theorem for flows which is stated below.

Theorem 2 (i) *If* $\mathbf{B}$ *is an M-flow, then* $\alpha(\mathbf{B})$ *is an M-FLOW.*

(ii) *If* $\mathbf{C}$ *is an M-FLOW, then* $\beta(\mathbf{C})$ *is an M-flow.*

(iii) *If* $\mathbf{B}$ *is an M-flow then* $\beta(\alpha(\mathbf{B})) = \mathbf{B}$, *and if* $\mathbf{C}$ *is an M-FLOW then* $\alpha(\beta(\mathbf{C})) = \mathbf{C}$.

The difficult part of the proof is the passing from (BI)FLOWS to (bi)flows. In order to make the proof easier to understand we insert a section with deductions of certain identities that are valid in a (BI)FLOW.

In the sequal we shall use two types of duality, briefly presented here.

Duality. We denote by a° the opposite of the word $a \in M$. Let $t \in B(a,b)$ be a term written with sum, extended feedback and identities.

The ${}^\circ$*-dual term* $t^\circ \in B(a^\circ, b^\circ)$ is obtained by using the following inductive procedure:

the ${}^\circ$*-dual term* of a variable $x \in B(a,b)$ is another variable $x^\circ \in B(a^\circ, b^\circ)$, and

$$(f+g)^\circ = g^\circ + f^\circ; (f \uparrow^{a(u)b}_{c(u)d})^\circ = f^\circ \uparrow^{b^\circ(u^\circ)a^\circ}_{d^\circ(u^\circ)c^\circ}; (\mathrm{I}_a)^\circ = \mathrm{I}_{a^\circ}$$

The ${}_\circ$*-dual term* $t_\circ \in B(b,a)$ is obtained by using the rules:
the ${}_\circ$*-dual term* of a variable $x \in B(a,b)$ is another variable $x_\circ \in B(b,a)$, and

$$(f+g)_\circ = f_\circ + g_\circ; (f \uparrow^{a(u)b}_{c(u)d})_\circ = f_\circ \uparrow^{c(u)d}_{a(u)b}; (\mathrm{I}_a)_\circ = \mathrm{I}_a$$

Lemma 1 (${}^\circ$-duality). *If E is an identity which is valid in every M-FLOW, then E° is an identity which is valid in every M-FLOW.*
(${}_\circ$-duality) *If E is an identity which is valid in every M-FLOW, then $E_\circ$ is an identity which is valid in every M-FLOW.*

Proof. It is enough to see that the ${}^\circ$-dual (resp. ${}_\circ$-dual) identity corresponding to a FLOW-axiom in Table 2 is also in Table 2, and the rules for deduction of valid identities (i.e. Manysorted Equational Logic) are invariant under ${}^\circ$-duality (resp. under ${}_\circ$-duality). □

Clearly, for every E we have $(E^\circ)_\circ = (E_\circ)^\circ$; in the sequel we shall denote it simply by $E^\circ_\circ$. From Lemma 3 it follows that:

Corollary (${}^\circ_\circ$-duality). *If E is an identity which is valid in every M-FLOW, then $E^\circ_\circ$ is an identity which is valid in every M- FLOW.* □

In order to simplify the calculation we shall use the following convention.

Convention. The writing of 0, which denotes the neutral element of the underlying monoid M, is usualy omitted.

If we are given two strings of letters, namely $\alpha = a_0(u_{\sigma(1)})a_1 \ldots a_{n-1}(u_{\sigma(n)})a_n$ and $\beta = b_0(u_1)b_1 \ldots b_{n-1}(u_n)b_n$ where each letter denotes an element in M, if moreover $\sigma : [n] \longrightarrow [n]$[2] is a bijection and $u_1, \ldots, u_n$ are all *distinct* letters, then by $\uparrow^\alpha_\beta$ we mean the multiple feedbackation computed, say, from right to left with respect to β. (By axioms (F2a) & (F2b) the order in which the feedbacks are computed is without importance.)For instance

$$f \uparrow^{(a)(v)b'(u)c'}_{(a)(u)b(v)c} (= f \uparrow^{0(a)0(v)b'(u)c'}_{0(a)0(u)b(v)c}) \text{ means } f \uparrow^{a(v)b'uc'}_{aub(v)c} \uparrow^{ab'(u)c'}_{a(u)bc} \uparrow^{(a)b'c'}_{(a)bc} .$$

This rule is ambiguous when some letters in $u_1, \ldots, u_n$ are equal. In that case we use numbers to indicate the correct feedbacks (and *not* the order in which the feedbacks are computed, which is without importance). For instance

$$f \uparrow^{(2,u)b(1,u)a}_{(1,u)d(2,u)c} \text{ means } f \uparrow^{(u)bua}_{ud(u)c} \uparrow^{b(u)a}_{(u)dc} .$$

□

With this convention the axioms (F0b) and (F0b$^\circ$), which are equivalent in the presence of (F2a), may be easily written as $f \uparrow^{a(uv)b}_{c(uv)d} = f \uparrow^{a(u)(v)b}_{c(u)(v)d}$.

Lemma 2 *In the presence of (F0-2) the axioms (F3) & (F3 ${}^\circ_\circ$) are equivalent to (F3$_\circ$) & (F3 $^\circ$).*

[2] $[n] = \{1, 2, \ldots, n\}$

Table 3: These identities are valid in a FLOW

$F5\ (\mathrm{I}_a + f)\uparrow^{a(a)b}_{(a)c} = f$	$F5^\circ\quad f = (f + \mathrm{I}_a)\uparrow^{b(a)a}_{c(a)}$
$F5_\circ\ (\mathrm{I}_a + f)\uparrow^{(a)c}_{a(a)b} = f$	$F5^\circ_\circ\quad f = (f + \mathrm{I}_a)\uparrow^{c(a)}_{b(a)a}$
$F6\ (\mathrm{I}_u + f)\uparrow^{(2,u)b(1,u)a}_{(1,u)d(2,u)c} = f\uparrow^{b(u)a}_{d(u)c}$	$F6^\circ\quad f\uparrow^{a(u)b}_{c(u)d} = (f + \mathrm{I}_u)\uparrow^{a(1,u)b(2,u)}_{c(2,u)d(1,u)}$

$$F7\ (= F7^\circ_\circ)(f+\mathrm{I}_u+g)\uparrow^{a(1,u)v(2,u)w}_{b(1,u)c(2,u)d} = (f+g)\uparrow^{av(u)w}_{b(u)cd}\ \text{ for } f : a \to buc, g : vuw \to d$$

$$F7^\circ\ (= F7_\circ)(g+f)\uparrow^{w(u)va}_{dc(u)b} = (g+\mathrm{I}_u+f)\uparrow^{w(2,u)v(1,u)a}_{d(2,u)c(1,u)b}\ \text{ for} f : a \to cub, g : wuv \to d$$

$$F8\quad (\mathrm{I}_b + f)\uparrow^{ba(b)}_{(b)c} = (f + \mathrm{I}_a)\uparrow^{(a)ba}_{c(a)}$$

$$F8_\circ\quad (\mathrm{I}_b + f)\uparrow^{(b)c}_{ba(b)} = (f + \mathrm{I}_a)\uparrow^{c(a)}_{(a)ba}$$

Proof. Suppose (F3) and $(F3^\circ_\circ)$ hold. Then $(F3^\circ)$ may be proved as follows:

$(f + \mathrm{I}_a)\uparrow^{(a)a}_{b(a)} =$ (by $F3^\circ_\circ$, then $F3)(\mathrm{I}_a + ((f + \mathrm{I}_a)\uparrow^{(a)a}_{b(a)} + \mathrm{I}_b)\uparrow^{a(b)}_{(b)b})\uparrow^{a(a)}_{(a)b}$

$= (\mathrm{I}_a + f + \mathrm{I}_{ab})\uparrow^{a(1,a)(2,a)(b)}_{(2,a)(b)(1,a)b} = (\mathrm{I}_a + f + \mathrm{I}_{ab})\uparrow^{aa(ab)}_{(ab)ab}\uparrow^{a(a)}_{(a)b} =$ (by $F3^\circ_\circ$, then $F3)f$.

We have proved that

(x) (F3) & $F3^\circ_\circ \Longrightarrow F3^\circ$.

By applying ${}^\circ_\circ$-duality to (x) it follows that $(F3^\circ_\circ)$& (F3) $\Longrightarrow F3_\circ$. By applying ${}^\circ$-duality to (x) it follows that $(F3^\circ)\&(F3_\circ) \Longrightarrow (F3)$, and by applying ${}_\circ$-duality to (x) it follows that $(F3_\circ)\&(F3^\circ) \Longrightarrow (F3^\circ_\circ)$. □

Lemma 3 *The identities in Table 3 are valid in a FLOW.*

Proof. Proof of (F5):

$(\mathrm{I}_a + f)\uparrow^{a(a)b}_{(a)c} =$ (by $F3^\circ$, and $F3_\circ)[\mathrm{I}_c + ((\mathrm{I}_a + f)\uparrow^{a(a)b}_{(a)c} + \mathrm{I}_{ab})\uparrow^{(ab)ab}_{c(ab)}]\uparrow^{(c)ab}_{c(c)}$

$= (\mathrm{I}_{ca} + f + I_{ab})\uparrow^{(c)(2,a)(1,a)(b)ab}_{c(1,a)(c)(2,a)(b)} = [(\mathrm{I}_{ca} + f + \mathrm{I}_a)\uparrow^{(ca)aba}_{ca(ca)} + \mathrm{I}_b]\uparrow^{(ab)ab}_{c(ab)}$

=(by $F3_\circ$ and $F3^\circ$) f.

Proof of (F6):

$(\mathrm{I}_u + f)\uparrow^{(2,u)b(1,u)a}_{(1,u)d(2,u)c} =$ (by $F5)(\mathrm{I}_b + (\mathrm{I}_u + f)\uparrow^{(2,u)b(1,u)a}_{(1,u)d(2,u)c})\uparrow^{b(b)a}_{(b)dc}$

$= (\mathrm{I}_{bu} + f)\uparrow^{b(2,u)(b)(1,u)a}_{(b)(1,u)d(2,u)c} = (\mathrm{I}_{bu} + f)\uparrow^{bu(bu)a}_{(bu)duc}\uparrow^{b(u)a}_{d(u)c} =$ (by $F5) f\uparrow^{b(u)a}_{d(u)c}$.

Proof of (F7):

$(f + \mathrm{I}_u + g)\uparrow^{a(1,u)v(2,u)w}_{b(1,u)c(2,u)d} =$ (by $F6^\circ)(f + \mathrm{I}_u + g + \mathrm{I}_u)\uparrow^{a(3,u)v(2,u)w(4,u)}_{b(4,u)c(2,u)d(3,u)}$

$= (f + (\mathrm{I}_u + g + \mathrm{I}_u)\uparrow^{(3,u)v(2,u)wu}_{(2,u)d(3,u)})\uparrow^{avw(4,u)}_{b(4,u)cd}$

$=$ (by $F6)(f + (g + \mathrm{I}_u)\uparrow^{v(5,u)wu}_{d(5,u)})\uparrow^{avw(4,u)}_{b(4,u)cd}$

$= (f + g + \mathrm{I}_u)\uparrow^{av(5,u)w(4,u)}_{b(4,u)cd(5,u)} =$ (by $F6^\circ)(f + g)\uparrow^{av(u)w}_{b(u)cd}$.

Proof of (F8):

$(\mathrm{I}_b + f)\uparrow^{ba(b)}_{(b)c} =$ (by $F5^\circ)((\mathrm{I}_b + f)\uparrow^{ba(b)}_{(b)c} + \mathrm{I}_a)\uparrow^{b(a)a}_{c(a)} = (\mathrm{I}_b + f + \mathrm{I}_a)\uparrow^{b(a)(b)a}_{(b)c(a)}$

$= (\mathrm{I}_b + (f + I_a)\uparrow^{(a)ba}_{c(a)})\uparrow^{b(b)a}_{(b)c} =$ (by $F5)(f + \mathrm{I}_a)\uparrow^{(a)ba}_{c(a)}$.

The proof af the remaining identities in Table 3 may be obtained from these ones by using ${}^\circ$-duality, ${}_\circ$- duality, or ${}^\circ_\circ$-duality. □

Proof of Theorem 1.(i) (resp. of Theorem 2.(i)). Clearly the identities in Table 2 hold in the algebras of flowchart schemes (resp. of flowchart scheme representations), hence by Theorem 8.2 in [CS88b] (resp. by Theorem 2.b.5. in [CS87a]) they hold in every biflow (resp. flow). Hence every biflow (resp. flow) is a BIFLOW (resp. FLOW). Direct deductions are given in [CS89]. □

Lemma 4 *The following identities hold in a FLOW.*

(T1) $f \circ (\mathrm{I}_b + {}^c\mathrm{X}^d) = (f + \mathrm{I}_c) \uparrow^{a(c)}_{b(c)dc}$ *for* $f : a \to bcd$

$(T1^{\circ}_{\circ})$ $(\mathrm{I}_c + f) \uparrow^{cd(c)b}_{(c)a} = ({}_d\mathrm{X}_c + \mathrm{I}_b) \circ f$ *for* $f : dcb \to a$

(T2) $f \circ ({}^b\mathrm{X}^c + \mathrm{I}_d) = (\mathrm{I}_c + f) \uparrow^{(c)a}_{cb(c)d}$ *for* $f : a \to bcd$

$(T2^{\circ}_{\circ})$ $(f + \mathrm{I}_c) \uparrow^{d(c)bc}_{a(c)} = (\mathrm{I}_d + {}_c\mathrm{X}_b) \circ f$ *for* $f : dcb \to a$

"$\circ$" and "X" are those defined by formulas (2) and (3), respectively; ${}_a\mathrm{X}_b := {}^b\mathrm{X}^a$*).*

Proof. First we prove (T1):

$$f \circ (\mathrm{I}_b + {}^c\mathrm{X}^d) = (f + \mathrm{I}_{bcdc}) \uparrow^{a(bcd)(c)}_{(bcd)b(c)dc} = ((f + \mathrm{I}_{bcd}) \uparrow^{a(bcd)}_{(bcd)bcd} + \mathrm{I}_c) \uparrow^{a(c)}_{b(c)dc}$$
$$= (\text{ by } F3^{\circ}_{\circ})(f + \mathrm{I}_c) \uparrow^{a(c)}_{b(c)dc} .$$

For (T2) note that

$$f \circ ({}^b\mathrm{X}^c + \mathrm{I}_d) = (f + \mathrm{I}_{bcbd}) \uparrow^{a(bc)(b)(d)}_{(bc)(d)(b)cbd} = (\text{ by } F7)(f + \mathrm{I}_{cbd}) \uparrow^{a(c)(b)(d)}_{(b)(c)(d)cbd}$$
$$= (\text{ by } F5_{\circ})(\mathrm{I}_c + (f + \mathrm{I}_{cbd}) \uparrow^{a(c)(b)(d)}_{(b)(c)(d)cbd}) \uparrow^{(c)a}_{c(c)bd}$$
$$= ((\mathrm{I}_c + f + \mathrm{I}_c) \uparrow^{(1,c)a(2,c)}_{cb(2,c)d(1,c)} + \mathrm{I}_{bd}) \uparrow^{a(bd)}_{c(bd)bd}$$
$$= (\text{ by } F6^{\circ} \text{ and } F5^{\circ}_{\circ})(\mathrm{I}_c + f) \uparrow^{(c)a}_{cb(c)d} .$$

The other identities $(T1^{\circ}_{\circ})$ and $(T2^{\circ}_{\circ})$ follows by using $^{\circ}_{\circ}$-duality. □

Proof of Theorem 2. (ii). Let $\mathbf{C} = (C, +, \uparrow^{a(u)b}_{c(u)d}, \mathrm{I}_a)$ be an M-FLOW. We have to show that $(C, +, \circ, \Uparrow^a, \mathrm{I}_a, {}^a\mathrm{X}^b)$ is an M-flow, where "$\circ$", "${}^a\mathrm{X}^b$", and "$\Uparrow^a$" are those defined in $\mathbf{C}$ by formulas (2), (3), and (4), respectively. That is, we have to verify the validity of the axioms (B1-5), (B6a,b,c), (B7a), (B8-16) in Table 1.

By using (F1), $(F1^{\circ})$, and (F2a) one may easily see that (B1) is valid, and by using (F3) and $(F3^{\circ}_{\circ})$ it follows that (B2) is valid. (B3-5) are common axioms. For (B6a) suppose $f : a \longrightarrow b, g : b \longrightarrow c$ and $g' : b' \longrightarrow c'$. Then

$$(f + \mathrm{I}_{b'}) \circ (g + g') = (f + \mathrm{I}_{b'} + g + g') \uparrow^{ab'(bb')}_{(bb')cc'}$$
$$= (\text{ by } F3^{\circ})((f + \mathrm{I}_{b'} + g + g') \uparrow^{ab'(bb')}_{(bb')cc'} + \mathrm{I}_{ab'}) \uparrow^{(ab')ab'}_{cc'(ab')}$$
$$= (f + (\mathrm{I}_{b'} + g + g' + I_{ab'}) \uparrow^{(1,b')b(2,b')ab'}_{(2,b')cc'a(1,b')}) \uparrow^{(a)(b)ab'}_{(b)cc'(a)}$$
$$= (\text{ by } F6)(f + (g + g' + \mathrm{I}_{ab'}) \uparrow^{b(b')ab'}_{cc'a(b')}) \uparrow^{(a)(b)ab'}_{(b)cc'(a)}$$
$$= [((f + g) \uparrow^{a(b)}_{(b)c} + g') + \mathrm{I}_{ab'}] \uparrow^{(ab')ab'}_{cc'(ab')} = (\text{ by } F3^{\circ}) f \circ g + g' = f \circ g + \mathrm{I}_{b'} \circ g'.$$

A proof of (F6b) may be obtained from the above proof of (F6a) by using $^{\circ}_{\circ}$-duality.

In order to prove the axioms for block transpositions we use the identities (T1), $(T1^{\circ}_{\circ})$ and (T2) in Lemma 6. For (B6c) note that

$$(\mathrm{I}_{ab} + f) \circ ({}^a\mathrm{X}^b + I_d) = (\text{ by } T2)(\mathrm{I}_b + \mathrm{I}_{ab} + f) \uparrow^{(b)abc}_{ba(b)d}$$
$$= \mathrm{I}_{bab} \uparrow^{(b)ab}_{ba(b)} + f = (\text{ by } F8)\mathrm{I}_{aba} \uparrow^{ab(a)}_{(a)ba} + f = {}^a\mathrm{X}^b + f = \mathrm{I}_{ab} \circ {}^a\mathrm{X}^b + f \circ \mathrm{I}_d.$$

For (B7a) note that

$${}^{c}\mathrm{X}^{a} \circ (f + \mathrm{I}_c) \circ {}^{b}\mathrm{X}^{c} = (\text{ by } T1)\, {}^{c}\mathrm{X}^{a} \circ (f + \mathrm{I}_c + \mathrm{I}_b) \uparrow^{ac(b)}_{(b)cb}$$
$$= (\text{ by } T1^{\circ}_{\circ})(\mathrm{I}_c + (f + \mathrm{I}_{cb}) \uparrow^{ac(b)}_{(b)cb}) \uparrow^{ca(c)}_{(c)cb} = ((\mathrm{I}_c + f) + I_{cb}) \uparrow^{ca(cb)}_{(cb)cb} = (\text{ by } F3^{\circ}_{\circ})\mathrm{I}_c + f.$$

The proof of (B8) is obvious. For (B9) note that

$$({}^{a}\mathrm{X}^{b} + \mathrm{I}_c) \circ (\mathrm{I}_b + {}^{a}\mathrm{X}^{c}) = (\text{ by } T1)({}^{a}\mathrm{X}^{b} + \mathrm{I}_c + \mathrm{I}_a) \uparrow^{abc(a)}_{b(a)ca}$$
$$= (\mathrm{I}_{aba} + \mathrm{I}_{ca}) \uparrow^{ab(1,a)c(2,a)}_{(1,a)b(2,a)ca} = (\text{ by } F7)\mathrm{I}_{abca} \uparrow^{abc(a)}_{(a)bca} = {}^{a}\mathrm{X}^{bc}.$$

The right feedback $\_ \Uparrow^{a}$ is defined by formula (4), i.e.

$$f \Uparrow^{a} = f \uparrow^{b(a)}_{c(a)} \text{ for } f : ba \longrightarrow ca.$$

Axiom (B10) is valid. Indeed, if $f : d \longrightarrow b$ and $g : ba \longrightarrow ca$, then

$$((f + \mathrm{I}_a) \circ g) \Uparrow^{a} = (f + \mathrm{I}_a + g) \uparrow^{da(ba)}_{(ba)ca} \uparrow^{d(a)}_{c(a)} = (f + (\mathrm{I}_a + g) \uparrow^{(2,a)b(1,a)}_{(1,a)c(2,a)}) \uparrow^{d(b)}_{(b)c}$$
$$= (\text{ by } F6)(f + g \uparrow^{b(a)}_{c(a)}) \uparrow^{d(b)}_{(b)c} = f \circ (g \Uparrow^{a}).$$

The proof of (B11) is similar. Axiom (B12) is a particular case of $(F1^{\circ})$. For (B13) suppose $f : cba \longrightarrow dab$. Then

$$(f \circ (\mathrm{I}_d +{}^{a} \mathrm{X}^{b})) \Uparrow^{ba} = (\text{ by } T1)(f + \mathrm{I}_a) \uparrow^{cba(a)}_{d(a)ba} \uparrow^{c(ba)}_{d(ba)} = (\text{ by } F6^{\circ}) f \uparrow^{c(b)(a)}_{d(a)(b)},$$

and similarly

$$((\mathrm{I}_c +{}^{a} \mathrm{X}^{b}) \circ f) \Uparrow^{ab} = (\text{ by } T2^{\circ}_{\circ})(f + \mathrm{I}_b) \uparrow^{c(b)ab}_{dab(b)} \uparrow^{c(ab)}_{d(ab)} = (\text{ by } F6^{\circ}) f \uparrow^{c(b)(a)}_{d(a)(b)},$$

hence (B13) is valid. Axiom (B14) is a particular case of $(F0b^{\circ})$ and (B15) is (F4). For (B16) note that

$${}^{a}\mathrm{X}^{a} \Uparrow^{a} = \mathrm{I}_{aaa} \uparrow^{aa(a)}_{(a)aa} \uparrow^{a(a)}_{a(a)} = (\mathrm{I}_{aa} + \mathrm{I}_a) \uparrow^{a(1,a)(2,a)}_{(2,a)a(1,a)} = (\text{ by } F6^{\circ})\mathrm{I}_{aa} \uparrow^{a(a)}_{(a)a} = (\text{ by } F3)\mathrm{I}_a.$$

□

Proof of Theorem 2. (iii). Suppose $\mathbf{B} = (B, +, \cdot, \uparrow^{a}, \mathrm{I}_a, {}^{a}\underline{\mathrm{X}}^{b})$ is an M-flow. The new composition "∘" in $\beta(\alpha(\mathbf{B}))$ acts as follows

$$f \circ g = (f + g) \Uparrow^{a(b)}_{(b)c} = [(\mathrm{I}_a + {}^{0}\underline{\mathrm{X}}^{b}) \cdot (f + g) \cdot (\mathrm{I}_0 + {}^{b}\underline{\mathrm{X}}^{c})] \uparrow^{b}$$
$$= ((f + g) \cdot {}^{b}\underline{\mathrm{X}}^{c}) \uparrow^{b} = f \cdot [(\mathrm{I}_b + g) \cdot {}^{b}\underline{\mathrm{X}}^{c}] \uparrow^{b} = f \cdot ({}^{b}\underline{\mathrm{X}}^{b} \cdot (g + \mathrm{I}_b)) \uparrow^{b}$$
$$= f \cdot ({}^{b}\underline{\mathrm{X}}^{b} \uparrow^{b}) \cdot g = f \cdot \mathrm{I}_b \cdot g = f \cdot g.$$

Hence $f \circ g = f \cdot g$.

The new block transposition ${}^{a}\mathrm{X}^{b}$ in $\beta(\alpha(\mathbf{B}))$ is

$${}^{a}\mathrm{X}^{b} = \mathrm{I}_{aba} \Uparrow^{ab(a)}_{(a)ba} = [(\mathrm{I}_{ab} + {}^{0}\underline{\mathrm{X}}^{a}) \cdot \mathrm{I}_{aba} \cdot (\mathrm{I}_0 + {}^{a}\underline{\mathrm{X}}^{ba})] \uparrow^{a}$$
$$= {}^{a}\underline{\mathrm{X}}^{ba} \uparrow^{a} = [({}^{a}\underline{\mathrm{X}}^{b} + \mathrm{I}_a) \cdot (\mathrm{I}_b + {}^{a}\underline{\mathrm{X}}^{a})] \uparrow^{a} = {}^{a}\underline{\mathrm{X}}^{b} \cdot (\mathrm{I}_b + {}^{a}\mathrm{X}^{a} \uparrow^{a}) = {}^{a}\underline{\mathrm{X}}^{b} \cdot (\mathrm{I}_b + \mathrm{I}_a) = {}^{a}\underline{\mathrm{X}}^{b}.$$

Hence ${}^{a}\mathrm{X}^{b} = {}^{a}\underline{\mathrm{X}}^{b}$.

The new right feedbackation $\_ \Uparrow^{a}$ in $\beta(\alpha(\mathbf{B}))$ acts as follows. For $f : ba \rightarrow ca$, $f \Uparrow^{a} = [(\mathrm{I}_b + {}^{0}\underline{\mathrm{X}}^{a}) \cdot f \cdot (\mathrm{I}_c + {}^{a}\underline{\mathrm{X}}^{0})] \uparrow^{a} = f \uparrow^{a}$.
Hence $\_ \Uparrow^{a} = \_ \uparrow^{a}$.

Conversely, suppose $\mathbf{C} = (C, +, \uparrow^{a(u)b}_{c(u)d}, \mathrm{I}_a)$ is an M-FLOW. The new extended feedbackation $\Uparrow^{a(u)b}_{c(u)d}$ in $\alpha(\beta(\mathbf{C}))$ acts as follows. For $f : aub \longrightarrow cud$

$$f \Uparrow^{a(u)b}_{c(u)d} = [(\mathrm{I}_a + {}^{b}\underline{\mathrm{X}}^{u}) \circ f \circ (\mathrm{I}_c + {}^{u}\underline{\mathrm{X}}^{d})] \Uparrow^{u}$$
$$= (\text{ by } T1)[(\mathrm{I}_a + {}^{b}\mathrm{X}^{u}) \circ (f + \mathrm{I}_u) \uparrow^{aub(u)}_{c(u)du}] \Uparrow^{u}$$
$$= (\text{ by } T2^{\circ}_{\circ})[(f + \mathrm{I}_u) \uparrow^{aub(u)}_{c(u)du} + \mathrm{I}_u] \uparrow^{a(u)bu}_{cdu(u)} \uparrow^{ab(u)}_{cd(u)}$$
$$= (\text{ by } F6^{\circ})(f + \mathrm{I}_u) \uparrow^{aub(u)}_{c(u)du} \uparrow^{a(u)b}_{cd(u)} = (\text{ by } F6^{\circ}) f \uparrow^{a(u)b}_{c(u)d}.$$

Hence $\Uparrow^{a(u)b}_{c(u)d} = \uparrow^{a(u)b}_{c(u)d}$. □

Table 4: These axioms define a scalar BIFLOW, while the subset of all the axioms denoted by SS.. or SF.. define a scalar FLOW

$SS1$	$f+(g+h)=(f+g)+h$		
$SS2$	$f+\mathrm{I}_s\uparrow^1_1=f$	$SS2^\circ$	$f=\mathrm{I}_s\uparrow^1_1+f$
$SF1$	$f\uparrow^i_j+g=(f+g)\uparrow^i_j$	$SF1^\circ$	$(g+f)\uparrow^{\lvert c\rvert+i}_{\lvert d\rvert+j}=g+f\uparrow^i_j$ for $g:c\longrightarrow d$
$SF2a$	$f\uparrow^{\lvert a\rvert+1}_{\lvert a'\rvert+1}\uparrow^{\lvert ab\rvert+1}_{\lvert a'b'\rvert+1}=f\uparrow^{\lvert ab\rvert+2}_{\lvert a'b'\rvert+2}\uparrow^{\lvert a\rvert+1}_{\lvert a'\rvert+1}$		for $f:asbtc\longrightarrow a'sb'tc'$
$SF2b$	$f\uparrow^{\lvert a\rvert+1}_{\lvert a'b'\rvert+2}\uparrow^{\lvert ab\rvert+1}_{\lvert a'\rvert+1}=f\uparrow^{\lvert ab\rvert+2}_{\lvert a'\rvert+1}\uparrow^{\lvert a\rvert+1}_{\lvert a'b'\rvert+1}$		for $f:asbtc\longrightarrow a'tb'sc'$
$SF3$	$(\mathrm{I}_{s_1}+\ldots\mathrm{I}_{s_n}+f)\uparrow^{n+n}_n\uparrow^{n+n-1}_{n-1}\ldots\uparrow^{n+1}_1=f$		for $f:s_1\ldots s_n\longrightarrow a$
$SF3^\circ_\circ$	$f=(f+\mathrm{I}_{s_n}+\ldots+\mathrm{I}_{s_1})\uparrow^{\lvert a\rvert+n}_n\uparrow^{\lvert a\rvert+n-1}_{n-1}\ldots\uparrow^{\lvert a\rvert+1}_1$		for $f:a\longrightarrow s_n\ldots s_1$
SP	$(f+g)\uparrow^{\lvert a\rvert+n}_n\uparrow^{\lvert a\rvert+n-1}_{n-1}\ldots\uparrow^{\lvert a\rvert+1}_1=(g+f)\uparrow^n_{\lvert b\rvert+n}\uparrow^{n-1}_{\lvert b\rvert+n-1}\cdots\uparrow^1_{\lvert b\rvert+1}$		for $f:a\longrightarrow s_1\ldots s_n, g:s_1\ldots s_n\longrightarrow b$

It has to be noted that a biflow is a flow fulfilling (B7). Indeed, (B6) may be proved by using (B7) as follows: if $f':a'\longrightarrow b'$ and $g:b\longrightarrow c$, then

$g+f'=(\text{ by }B7)^b\mathrm{X}^{a'}(f'+g)^{b'}\mathrm{X}^c=(\text{ by }B6a)^b\mathrm{X}^{a'}(f'+\mathrm{I}_b)(\mathrm{I}_{b'}+g)^{b'}\mathrm{X}^c$
$=(\text{ by }B7a)(\mathrm{I}_b+f')^b\mathrm{X}^{b'}\cdot^{b'}\mathrm{X}^b(g+\mathrm{I}_{b'})=(\text{ by }B7a)(\mathrm{I}_b+f')(g+\mathrm{I}_{b'})$

therefore if moreover $f:a\longrightarrow b$ and $g':b'\longrightarrow c'$, then

$(f+f')(g+g')=(\text{ by }B6a)(f+\mathrm{I}_{a'})(\mathrm{I}_b+f')(g+\mathrm{I}_{b'})(\mathrm{I}_c+g')$
$=(f+\mathrm{I}_a)(g+f')(\mathrm{I}_c+g')=(\text{ by }B6a)(fg+f')(\mathrm{I}_c+g')=(\text{ by }B6b)fg+f'g'.$

Proof of Theorem 1. The proof of this theorem is based on Theorem 2 and we shall use the notation in the above proof of Theorem 2. Suppose **B** is an M-biflow. We still have to show that $\Uparrow^{a(u)b}_{c(u)d}$ fulfils axiom (P). Indeed, if $f:a\longrightarrow b$ and $g:b\longrightarrow c$ then

$(f+g)\Uparrow^{a(b)}_{(b)c}=[(\mathrm{I}_a+{}^0\underline{\mathrm{X}}^b)(f+g)(\mathrm{I}_0+{}^b\underline{\mathrm{X}}^c)]\uparrow^b=((f+g)^b\underline{\mathrm{X}}^c)\uparrow^b$
$=(\text{ by }B7)({}^a\underline{\mathrm{X}}^b(g+f))\uparrow^b=[(\mathrm{I}_0+{}^a\underline{\mathrm{X}}^b)(g+f)(\mathrm{I}_c+{}^b\underline{\mathrm{X}}^0)]\uparrow^b=(g+f)\Uparrow^{(b)a}_{c(b)}.$

(ii) Suppose **C** is an M-BIFLOW. We still have to show that axiom (B7) holds. Indeed, if $f:a\longrightarrow b$ and $g:c\longrightarrow d$ then

${}^c\mathrm{X}^a\circ(f+g)\circ{}^b\mathrm{X}^d=(\text{ by }T1\text{ and }T1^\circ_\circ)(\mathrm{I}_c+f+g+\mathrm{I}_b)\uparrow^{ca(c)(b)}_{(c)(b)db}$
$=[(\mathrm{I}_c+f)+(g+\mathrm{I}_b)]\uparrow^{ca(cb)}_{(cb)db}=(\text{ by }P)[(g+\mathrm{I}_b)+(\mathrm{I}_c+f)]\uparrow^{(cb)ca}_{db(cb)}$
$=(\text{ by }F3^\circ)(g+\mathrm{I}_{bc}+f+\mathrm{I}_{ca})\uparrow^{(cb)(ca)ca}_{db(cb)(ca)}=(g+(\mathrm{I}_{bc}+f+\mathrm{I}_{ca})\uparrow^{(bc)aca}_{bc(bc)a})\uparrow^{(ca)ca}_{db(ca)}$
$=(\text{ by }F5_\circ)(g+f+\mathrm{I}_{ca})\uparrow^{(ca)ca}_{db(ca)}=(\text{ by }F3^\circ)g+f.$ □

Since the monoid M is a free monoid S^*, feedbackation may be restricted to letters. More precisely a

Scalar Extended Feedbackation is a family of operations

$\uparrow^i_j: B(s_1\ldots s_m,t_1\ldots t_n)\longrightarrow B(s_1\ldots s_{i-1}s_{i+1}\ldots s_m,t_1\ldots t_{j-1}t_{j+1}\ldots t_n),$

where $s_1,\ldots,s_m,t_1,\ldots,t_n\in S, i\in[m], j\in[n]$ are such that $s_i=t_j$.

The resulting axiom system is presented in Table 4. In this table the letters s ant t range over S, while the others over S^*, and $\lvert a\rvert$ denotes the lenght of a word $a\in S^*$. Axioms (SS1) and (SS2 & $SS2^\circ$) show the sum is associative with neutral element $\mathrm{I}_s\uparrow^1_1$. Axioms (SF1 & $SF1^\circ$) show the feedback commutes with sum. Axioms (SF3 & $SF3^\circ_0$) show identities behave in a natural way, and finally axiom (SP) is a permutability axiom.

References

[Ba87] BARTHA M., A finite axiomatization of flowchart schemes, Acta Cybernetica 8 (1987), 203-217.

[BEW80a] BLOOM S. L., ELGOT C. C. and WRIGHT J. B., Solution of the iteration equation and extensions of the scalar iteration operation. SIAM Journal of Computing, 9, 1(1980), 26-45.

[BE85] BLOOM S.L. AND ÉSIK Z., Axiomatizing schemes and their behaviours, J. Comput. System Sci. 31 (1985), 375- 393.

[CG84] CAZANESCU V.E. AND GRAMA S., "On the definition of M-flowcharts", Preprint Series in Mathematics No. 56/1984; also in: An. Univ. "Al. I. Cuza" Iasi, Mat XXXIII, 4 (1987), 311-320.

[CS87a] CAZANESCU V.E. AND STEFANESCU GH., " A Formal Representation of Flowchart Schemes," Preprint Series in Mathematics No. 22/June 1987, INCREST, Bucharest; also in: An.Univ. Bucuresti, Mat.-Inf. XXXVII, 2 (1988), 33-51.

[CS87b] CAZANESCU V.E. AND STEFANESCU GH., "Towards a New Algebraic Foundation of Flowchart Scheme Theory," Preprint Series in Mathematics No. 43/December 1987, INCREST, Bucharest; Fundamenta Informaticae XIII (1990), 171-210.

[CS88a] CAZANESCU V.E. AND STEFANESCU GH., "Feedback, Iteration and Repetition," Preprint Series in Mathematics No. 42/August 1988, INCREST, Bucharest.

[CS88b] CAZANESCU V.E. AND STEFANESCU GH., "A Formal Representation of Flowchart Schemes II," Preprint Series in Mathematics No.60/November 1988, INCREST, Bucharest; also in: Stud. Cerc. Mat 41 (1989), 151-167.

[CS89] CAZANESCU V.E. AND STEFANESCU GH., "An axiom system for biflow using summation, (extended) feedbackation, and identities", Preprint Series in Mathematics No. 19/May 1989, INCREST, Bucharest.

[CU82] CAZANESCU V.E. AND UNGUREANU C., " Again on advice on structure compilers and proving them correct", Preprint Series in Mathematics No. 75 November 1982, INCREST, Bucharest; published in two parts in Rev. Roumaine Math. Pures Appl. 33 (1988), 561-573 and 34 (1989), 281-302.

[El75] ELGOT C.C., Monadic computation and iterative algebraic theories. J. C. Shepherson, editor, Logic Colloquium 1973, Studies in Logic, volume 80. North Holland, Amsterdam, 1975.

[És80] ÉSIK Z. Identities in iterative and rational theories. Computational Linguistics and Computer Languages, 14(1980), 183-207.

[Pa87] PARROW J., "Synchronisation flow algebra," Report ECS-LFCS-87-35, Computer Science Department, University of Edinburgh, August 1987.

[St86a] STEFANESCU GH., An algebraic theory of flowchart schemes (extended abstract), in: Proceedings CAAP'86 (ed. P. Franchi-Zannettacci), LNCS 214, Springer Verlag 1986, 60-73.

[St86b] STEFANESCU GH., "Feedback Theories (A Calculus for Isomorphism Classes of Flowchart Schemes)," Preprint Series in Mathematics No. 24/April 1986, INCREST, Bucharest; Rev. Roumaine Math. Pures Appl. Vol. 35, No. 1(1990), 73-79.

(Received October 17, 1989)

Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1990

On the multidimensional vector bin packing

J. Csirik[*] J.B.G. Frenk M. Labbé[†] S. Zhang

Econometrisch Instituut, Erasmus Universiteit Rotterdam, P.O. Box 1738, 3000DR Rotterdam, the Netherlands

Abstract

The multidimensional vector bin packing problem consists in packing m-dimensional items into a minimum number of m-dimensional bins with unit capacity in each of the m dimensions in such a way that the sum of each coordinate of the items received by any bin is not larger than one. We improve the lower bound of the First-Fit-Decreasing heuristic when $m \geq 5$ and odd, and prove that this heuristic is optimal when $m = 2$ if each item has at least one coordinate larger than 1/2. Finally, if this last condition holds and $m \geq 3$, we show that the problem remains NP-hard.

1 Introduction

In the multidimensional vector bin packing problem (MDVPP), we are given a list

$$L = (x_1, x_2, \ldots, x_n)$$

of n items, where the items are vectors of form

$$(s_1(x_i), s_2(x_i), \ldots, s_m(x_i)), \; i = 1, 2, \ldots, n,$$

with $0 < s_j(x_i) \leq 1$, $j = 1, 2, \ldots, m$. Then, the problem is to pack the items into a minimum number of bins, of unit capacity in each dimension, in such a way that the vector sum of the items received by any bin does not exceed $(1, 1, \ldots, 1)$. Since this problem is a generalization of the classical one-dimensional bin packing problem, it is clearly NP-hard.

Garey et al.[1] analyse some heuristics to find an approximate solution to MDVPP. Specifically, they provide an exact worst-case bound for the First-Fit (FF) heuristic, but only lower and upper bounds for a variant of the First-Fit-Decreasing (FFD) method.

In this note, we improve Garey et al.'s lower bound. Then, for the special case where each item has at least one coordinate larger than 1/2, we show that FFD is optimal if $m = 2$ and that the problem remains NP-hard when $m \geq 3$.

[*]On leave from the Department of Computer Science, University of Szeged

[†]Fellow of the European Institute for Advanced Studies in Management, Brussels

2 Definitions and the lower bound for FFD

We denote the optimal number of bins for the list $L = (x_1, x_2, \ldots, x_n)$ by $OPT(L)$. For a heuristic algorithm A, we denote the number of bins used when applying A to L by $A(L)$. Let

$$R_A(k) = \max\{A(L)/OPT(L) : OPT(L) = k\}.$$

The asymptotic worst-case ratio of algorithm A is defined as

$$R_A = \limsup_{k \to \infty} R_A(k).$$

We consider the following generalization of the one-dimensional FFD algorithm.
Generalized First-Fit-Decreasing heuristic (GFFD)

Step 1 Reorder the list $L = (x_1, x_2, \ldots, x_n)$ in such a way that

$$s_{\max}(x_1) \geq s_{\max}(x_2) \geq \cdots \geq s_{\max}(x_n),$$

where $s_{\max}(x) = \max_{j=1,\ldots,m} s_j(x)$.

Step 2 Apply the FF heuristic to the ordered list (as for the one-dimensional case (cf.[1])).

Garey et al. [1] prove that for this heuristic

$$m + \frac{1}{m+2} - \frac{2}{m(m+1)(m+2)} \leq R_{GFFD} \leq m + \frac{1}{3}, \quad \text{if } m \geq 4.$$

For the special cases where $1 \leq m \leq 3$, they obtain slightly better bounds. Specifically, for $m = 1$, the exact ratio is $11/9$, and for $m = 2$ or 3, the lower bound is $m + 11/60$.

We now improve Garey et al.'s bound on R_{GFFD} for $m \geq 5$ and odd.

Lemma 1 *For $m \geq 5$ and odd,*

$$R_{GFFD} \geq m + \frac{1}{m+2} - \frac{1}{m(m+1)(m+2)}.$$

Proof. We use the following "bad" list (the first part is the same as given in Graham et al. [1]). Let k be an arbitrary positive integer which is a multiple of $m(m+1)(m+2)$. The list L is composed of m regions, the items in region i occur in L before the items in region $i+1$, $1 \leq i < m$. The items in region i are denoted by

$$x_{i,1}, x_{i,2}, \ldots, x_{i,q(i)}$$

where $q(i) = (i+1)(k-1)$ for $1 \leq i < m$, and $q(m) = (m+2)k$. Furthermore, let $0 < \varepsilon < k^{-4}$. We define the item coordinates as follows.
For $1 \leq i < m$,

$$s_i(x_{i,j}) = \begin{cases} \frac{1}{i+1} + \varepsilon t_{i,j}, & \text{if } 1 \leq j \leq i(k-1), \\ \frac{1}{i+1} - i\varepsilon t'_{i,j}, & \text{if } i(k-1) < j \leq q(i), \end{cases}$$

where $t_{i,j} = k - \lfloor \frac{j-1}{i} \rfloor, t'_{i,j} = j + 1 - i(k-1)$ and

$$s_l(x_{i,j}) = \varepsilon/m^2 \text{ for } 1 \le l \le m, l \ne i \text{ and } 1 \le j \le q(i).$$

For the items of the last region,

$$s_l(x_{m,j}) = \varepsilon/(2 \cdot m^2) \text{ for } 1 \le l \le m-1 \text{ and } 1 \le j \le q(m),$$

and

$$s_m(x_{m,j}) = \begin{cases} \frac{1}{m+1} + \frac{2}{m(m+1)(m+3)} - \frac{\varepsilon}{m+2}, & \text{if } 1 \le j \le k\frac{m-1}{2}, \\ \frac{1}{m+2} + \frac{1}{(m+1)(m+2)(m+3)} - \frac{\varepsilon}{m+2}, & \text{if } k\frac{m-1}{2} < j \le k\frac{m+1}{2}, \\ \frac{1}{m+3} + \frac{2}{m(m+1)(m+3)^2} - \frac{\varepsilon}{m+2}, & \text{if } k\frac{m+1}{2} < j \le q(m). \end{cases}$$

So, specifically, items in the first region have the following sizes:
$s(x_{1,1}) = (1/2 + k \cdot \varepsilon, \varepsilon/m^2, \varepsilon/m^2, \dots, \varepsilon/m^2)$,
$s(x_{1,2}) = (1/2 + (k-1) \cdot \varepsilon, \varepsilon/m^2, \varepsilon/m^2, \dots, \varepsilon/m^2)$,

$$\vdots$$

$s(x_{1,k-1}) = (1/2 + 2 \cdot \varepsilon, \varepsilon/m^2, \varepsilon/m^2, \dots, \varepsilon/m^2)$,

$s(x_{1,k}) = (1/2 - 2 \cdot \varepsilon, \varepsilon/m^2, \varepsilon/m^2, \dots, \varepsilon/m^2)$,
$s(x_{1,k+1}) = (1/2 - 3 \cdot \varepsilon, \varepsilon/m^2, \varepsilon/m^2, \dots, \varepsilon/m^2)$,

$$\vdots$$

$s(x_{1,2(k-1)}) = (1/2 - k \cdot \varepsilon, \varepsilon/m^2, \varepsilon/m^2, \dots, \varepsilon/m^2)$.

Accordingly, in the i-th region $(2 < i < m)$ (the "big" coordinates are in the i-th position):
$s(x_{i,1}) = s(x_{i,2}) = \dots = s(x_{i,i}) =$

$$= (\varepsilon/m^2, \dots, \varepsilon/m^2, \frac{1}{i+1} + k\varepsilon, \varepsilon/m^2, \dots, \varepsilon/m^2),$$

$s(x_{i,i+1}) = s(x_{i,i+2}) = \dots = s(x_{i,2i}) =$

$$= (\varepsilon/m^2, \dots, \varepsilon/m^2, \frac{1}{i+1} + (k-1)\varepsilon, \varepsilon/m^2, \dots, \varepsilon/m^2),$$

$$\vdots$$

$s(x_{i,(k-2)i+1}) = s(x_{i,(k-2)i+2}) = \dots = s(x_{i,(k-1)i}) =$

$$= (\varepsilon/m^2, \dots, \varepsilon/m^2, \frac{1}{i+1} + 2\varepsilon, \varepsilon/m^2, \dots, \varepsilon/m^2),$$

$s(x_{i,(k-1)i+1}) = (\varepsilon/m^2, \dots, \varepsilon/m^2, \frac{1}{i+1} - 2 \cdot i \cdot \varepsilon, \varepsilon/m^2, \dots, \varepsilon/m^2)$,

$$s(x_{i,(k-1)i+2}) = (\varepsilon/m^2,\ldots,\varepsilon/m^2,\tfrac{1}{i+1} - 3\cdot i\cdot\varepsilon,\varepsilon/m^2,\ldots,\varepsilon/m^2).$$

$$\vdots$$

$$s(x_{i,(k-1)(i+1)}) = (\varepsilon/m^2,\ldots,\varepsilon/m^2,\tfrac{1}{i+1} - k\cdot i\cdot\varepsilon,\varepsilon/m^2,\ldots,\varepsilon/m^2).$$

When applying GFFD to our list L, we may also partition the set of bins used into m subsets, each bin of subset i containing only items from region i of list L. For $1 \le i \le m-1$, we have $(k-1)$ bins in subset i, the l-th bin of them contains exactly i items with $s_i = \frac{1}{i+1}+(k+1-l)\varepsilon$ and one item with $s_i = \frac{1}{i+1}-i(k+1-l)\varepsilon$. With these items, the bin is full in the i-th dimension. So, we use $(m-1)(k-1)$ bins for the items in the $(m-1)$ first regions of L, and we can not pack items from the later regions in these bins, even if their i-th coordinate is just ε/m^2.
Concerning the items in the m-th region, their "large" coordinate is the last one and can be of three different types. It is easy to check that all of these items will be packed in some bin together with other items, the "large" coordinate of which being of the same type. Hence, we use $k\cdot\frac{m-1}{2}/m$ bins for the items with large coordinate of the first type, $k/(m+1)$ bins for the items of the second type and $k\cdot\frac{m+3}{2}/(m+2)$ for the third type. Consequently, the total number of bins used when applying GFFD to L is

$$(k-1)(m-1)+k\cdot(1+\frac{1}{m+2}-\frac{1}{m(m+1)(m+2)})$$

On the other hand, the optimal packing uses at most k bins. To see this, we provide the following packing with k bins. Each bin contains $m+2$ items from the last region, i.e. $(m-1)/2$ items of the first type, one item of the second type and $(m+3)/2$ items of the third type. With these items, the sum of item sizes at each bin is

$$(\frac{m+2}{2m^2}\varepsilon,\frac{m+2}{2m^2}\varepsilon,\ldots,\frac{m+2}{2m^2}\varepsilon,1-\varepsilon).$$

Moreover, each bin contains items from each of the other regions. Specifically:

- The first bin contains the first i items from each region i $(i<m)$, these are the items with size $s_i = 1/(i+1)+k\cdot\varepsilon$.
- The last bin contains the $((k-1)i+1)$-th item of each region $i<m$.
- Each remaining bin contains $i+1$ items from each subset i, i.e. i items with size $s_i = 1/(i+1)-t\cdot\varepsilon$ and one item with $s_i = 1/(i+1)+i\cdot(t-1)\cdot\varepsilon$ (for an appropriate t).

Note that for all bins and in each dimension, the capacity used by "large" coordinates is never bigger than $1-\varepsilon$ so that we leave place enough for "small" coordinates.
In conclusion, for our list L,

$$\frac{\text{GFFD}(L)}{\text{OPT}(L)} \ge \frac{1}{k}[(k-1)(m-1)+k(1+\frac{1}{m+2}-\frac{1}{m(m+1)(m+2)})]$$

$$= m+\frac{1}{m+2}-\frac{1}{m(m+1)(m+2)}-\frac{m-1}{k},$$

which can be made arbitrarily close to

$$m+\frac{1}{m+2}-\frac{1}{m(m+1)(m+2)}$$

by choosing k large.

□

3 The special case

We study here the MDVPP for lists $L=(x_1,x_2,\ldots,x_n)$ such that

$$s_{\max}(x_i)>1/2 \quad \text{for } i=1,2,\ldots,n. \tag{1}$$

A. *When $m=2$, we prove that GFFD is optimal.* The idea behind this result is that, in such a case, each bin can contain at most two items, a situation which also occurs in the classical bin packing when all items have a size larger than 1/3 and this case is known to be polynomial.

Lemma 2 *If $m=2$, then for all lists for which (1) holds,* GFFD*(L)*=OPT*(L)*.

Proof. Assume that the elements in L are ranked by decreasing value of their largest coordinate. Furthermore, let $L=L_1\cup L_2$ with $L_1=(x_1,x_2,\ldots,x_p)$ and $L_2=(y_1,y_2,\ldots,y_q)$ such that $s_{\max}(x_i)=s_1(x_i)$, $i=1,2,\ldots,p$ and $s_{\max}(y_i)=s_2(y_i)$, $i=1,2,\ldots,q$. Without loss of generality, we may assume that $L_1\cap L_2=\emptyset$.

Now, define a bipartitate graph $G=(L_1,L_2,E)$ where $(x_i,y_j)\in E$ iff $s_k(x_i)+s_k(y_j)\le 1$, $k=1,2$. Since each bin can contain at most two elements (one from L_1 and the other one from L_2), the optimal packing of L corresponds to maximum matching M^* of G. Hence, to prove that GFFD is optimal, we shall show that the matching M_{GFFD} corresponding to the GFFD solution is optimal in G, i.e. there exists no augmenting path with respect to M_{GFFD} in G (see e.g. Papadimitriou and Steiglitz [3]). At the end of this proof, such a heuristic solution and its corresponding matching are illustrated by an example.

To begin with, remark that if $x_i\in L_1$ and $y_j\in L_2$ are both free vertices for M_{GFFD}, then $(x_i,y_j)\notin E$, for otherwise they would have been put together in a bin when applying GFFD.

Now, assume, by contradiction, that there exists an alternating path with respect to M_{GFFD}. From the above remark, we know that such a path contains more that one edge, i.e. at least three edges. Let

$$P=\{x_{i_1},y_{j_1},x_{i_2},y_{j_2},\ldots,y_{j_{l-1}},x_{i_l},y_{j_l}\}$$

be a *minimal alternating path* with respect to M_{GFFD}. Hence, x_{i_1} and y_{j_l} are free vertices, $(x_{i_k},y_{j_k})\in E\setminus M_{\text{GFFD}}$ for $k=1,2,\ldots,l$ and $(x_{i_{k+1}},y_{j_k})\in M_{\text{GFFD}}$ for $k=1,2,\ldots,l-1$. (see Figure 1 in which the edges of M_{GFFD} are indicated in waved lines). Furthermore, we denote by $x_{i_h}\succ x_{i_k}$ the fact that x_{i_h} has been considered before x_{i_k} when applying GFFD (i.e. $s_1(x_{i_h})\ge s_1(x_{i_k})$). The same notation applies for items in L_2.

Claim. $x_{i_{k+1}}\succ x_{i_k}$ and $y_{i_{k+1}}\succ y_{i_k}$ for $k=1,2,\ldots,l-1$.

Proof. By induction.

a) $k=1$. From the definition of P, $(x_{i_1},y_{j_1})\in E\setminus M_{\text{GFFD}}$ and $(y_{j_1},x_{i_2})\in M_{\text{GFFD}}$, i.e. GFFD put y_{j_1} and x_{i_2} in the same bin. Since x_{i_1} fits also with y_{j_1} in a bin, this means that $x_{i_2}\succ x_{i_1}$.

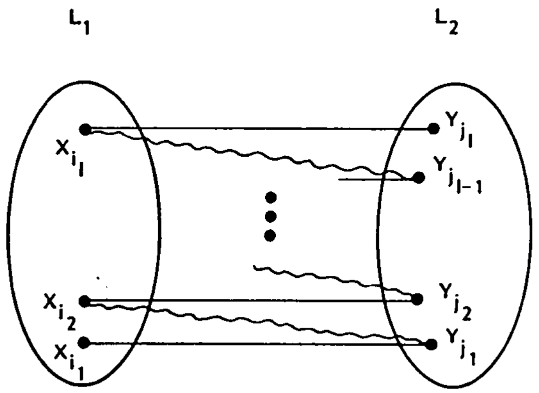

Figure 1: The minimal alternating path

Next, if $y_{j_2} \prec y_{j_1}$, we have that

$$s_1(x_{i_1}) + s_1(y_{j_2}) \le s_1(x_{i_2}) + s_1(y_{j_2}),$$

since $x_{i_2} \succ x_{i_1}$, and the right size is bounded above by 1, since $(x_{i_2}, y_{j_2}) \in E$; and

$$s_2(x_{i_1}) + s_2(y_{j_2}) \le s_2(x_{i_1}) + s_2(y_{j_1}),$$

since $y_{j_2} \prec y_{j_1}$, and the right size is bounded above by 1, since $(x_{i_1}, y_{j_1}) \in E$.

Hence $y_{j_2} \prec y_{j_1}$ implies that $(x_{i_1}, y_{j_2}) \in E$, which is impossible. Indeed, if $(x_{i_1}, y_{j_2}) \in E \setminus M_{\text{GFFD}}$, then P is not minimal. If $(x_{i_1}, y_{j_2}) \in M_{\text{GFFD}}$, then P contains a cycle. Consequently, $y_{j_2} \succ y_{j_1}$.

b) Assume $x_{i_{k-1}} \succ \ldots \succ x_{i_2} \succ x_{i_1}$ and $y_{i_{k-1}} \succ \ldots \succ y_{i_2} \succ y_{i_1}$. Since $y_{j_{k-1}}$ is not free and $y_{j_{k-1}} \succ y_{j_{k-2}}$ has been matched before $y_{j_{k-2}}$ when using GFFD. Because both $y_{j_{k-1}}$ and $y_{j_{k-2}}$ could have been matched with $x_{i_{k-1}}$, the item x_{i_k}, with which $y_{j_{k-1}}$ has been matched must be such that $x_{i_k} \succ x_{i_{k-1}}$.

Now, as in a), if $y_{j_k} \prec y_{j_{k-1}}$, we have that

$$s_1(x_{i_{k-1}}) + s_1(y_{j_k}) \le s_1(x_{i_k}) + s_1(y_{j_k}),$$

since $x_{i_k} \succ x_{i_{k-1}}$, and the right size is bounded above by 1, since $(x_{i_k}, y_{j_k}) \in E$; and

$$s_2(x_{i_{k-1}}) + s_2(y_{j_k}) \le s_2(x_{i_{k-1}}) + s_2(y_{j_{k-1}}),$$

since $y_{j_k} \prec y_{j_{k-1}}$, and the right size is bounded above by 1, since $(x_{i_{k-1}}, y_{j_{k-1}}) \in E$.

Hence $y_{j_k} \prec y_{j_{k-1}}$ implies that $(x_{i_{k-1}}, y_{j_k}) \in E$.

Then, if $(x_{i_{k-1}}, y_{j_l}) \in E \setminus M_{\text{GFFD}}$, P is not minimal and if $(x_{i_{k-1}}, y_{j_k}) \in M_{\text{GFFD}}$, M_{GFFD} contains two edge incident to $x_{i_{p-1}}$, which is impossible. Hence, $y_{j_k} \succ y_{j_{k-1}}$.

This completes the proof of the claim.

Finally, we know that the free item y_{j_l} of P which is also in L_2 is such that $y_{j_l} \succ y_{j_{l-1}}$. This is a contradiction, since, when applying GFFD, we considered it before $y_{j_{l-1}}$ and we did not matched it with x_{i_l}, though it was possible.

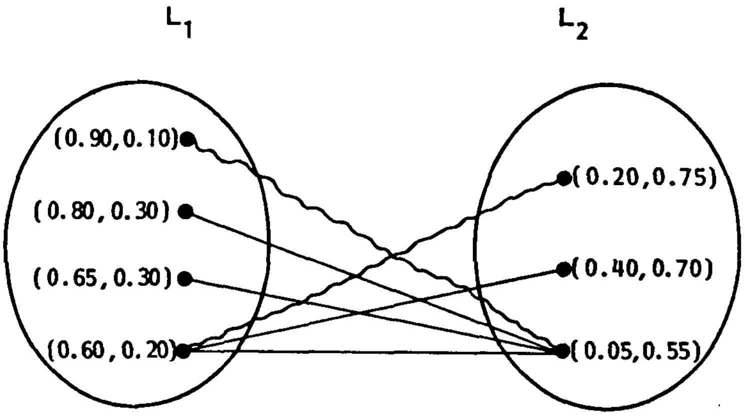

Figure 2: The graph and its maximum matching M_{GFFD} corresponding to the example.

Example: Consider the following list

$$L = \big((0.90,0.10),(0.80,0.30),(0.20,0.75),(0.40,0.70),(0.65,0.30),$$

$$(0.60,0.20),(0.05,0.55)\big).$$

When applying heuristic GFFD, we get GFFD(L)=5 and the corresponding bins are
$B_1 = \{(0.90,0.10)(0.05,0.55)\}$,
$B_2 = \{(0.80,0.30)\}$,
$B_3 = \{(0.20,0.75)(0.60,0.20)\}$,
$B_4 = \{(0.40,0.70)\}$,
$B_5 = \{(0.65,0.30)\}$.
The graph associated with L is presented in Figure 2 where the edges of the maximum matching M_{GFFD} are indicated in waved lines.

B. *When the dimension of the items in a list L is at least three,* MDVPP remains unfortunately NP-hard even if each item has at least one coordinate larger than 1/2.

To see this, we first define the decision version of the 3-dimensional vector bin packing problem for which condition (1) holds (we call it 3-DVPP with $s_{\max}$ large.)
3-DVPP with $s_{\max}$ large (P1)
INSTANCE: A finite set L of 3-dimensional nonnegative integer vectors

$$a_i = (s_1(a_i), s_2(a_i), s_3(a_i)), \quad i = 1,2,\dots,n.$$

A positive integer bin capacity B such that $\max(s_1(a_i), s_2(a_i), s_3(a_i)) > B/2$ for $i = 1,2,\dots,n$, and a positive integer K.

QUESTION: Is there a partition of L into disjoint sets $L_1, L_2, \dots, L_K$ such that $\sum_{a_i \in L_h} s_j(a_i) \le B$ for $j = 1,2,3$ and $h = 1,2,...,K$?

Lemma 3 *3-DVPP with $s_{\max}$ large is NP-complete.*

Proof. Clearly, this problem belongs to NP. To prove it is NP-complete we show that *NUMERICAL 3-DIMENSIONAL MATCHING* (which is NP-complete, (cf.[2], p.224)) reduces to a special case of our problem where $n = 3m$ and $K = m$.

NUMERICAL 3-DIMENSIONAL MATCHING (P2)

INSTANCE: Three disjoint sets X, Y and Z, each containing m elements, a nonnegative integer size $c(a)$ for each element $a \in X \cup Y \cup Z$, and a nonnegative integer bound B.

QUESTION: Can $X \cup Y \cup Z$ be partitioned inte m disjoint sets $L_1, L_2, \dots, L_m$ such that each L_i contains exactly one element from each set X, Y and Z such that for $i = 1, 2, \dots, m$:

$$\sum_{a \in L_i} c(a) = B?$$

We construct the instance of (P1) based on the instance of (P2) in the following way.

- For $a \in X$, define $s_1(a) = 2B/3, s_2(a) = 0,$ and $s_3(a) = c(a)/2$.
- For $a \in Y$, define $s_1(a) = 0, s_2(a) = 2B/3,$ and $s_3(a) = c(a)/2$.
- For $a \in Z$, define $s_1(a) = 0, s_2(a) = 0,$ and $s_3(a) = B/2 + c(a)/2$.

Now, consider a nontrivial instance of (P2), i.e. where $\max_{a \in X \cup Y \cup Z} c(a) \leq B$ and $\sum_{a \in X \cup Y \cup Z} c(a) = mB$. Hence, $c(a)/2 \leq B/2 < B$ for $a \in X \cup Y \cup Z$ and the reduced instance of (P2) is indeed an instance of (P1).

Assume now that the answer to the reduced instance is yes. Then, since each item has at least one coordinate larger than $B/2$, each set $L_1, L_2, \dots, L_m$ contains at most three items, i.e. at most one from X, one from Y and one from Z. Further, $n = 3m$ implies that each set $L_i, i = 1, 2, \dots m$ contains exactly three items, say $x_i \in X, y_i \in Y$ and $z_i \in Z$. Furthermore, we know that

$$s_3(x_i) + s_3(y_i) + s_3(z_i) \leq B \quad \text{for } i = 1, 2, \dots, m. \tag{2}$$

However,

$$\sum_{a \in X \cup Y \cup Z} s_3(a) = \sum_{a \in X} c(a)/2 + \sum_{a \in Y} c(a)/2 + \sum_{a \in Z} (B/2 + c(a)/2)$$
$$= \frac{1}{2} \sum_{a \in X \cup Y \cup Z} c(a) + \frac{B}{2} \mid Z \mid = mB/2 + mB/2 = mB.$$

In consequence, (2) must be satisfied as an equality, i.e.

$$B = s_3(x_i) + s_3(y_i) + s_3(z_i) = c(x_i)/2 + c(y_i)/2 + B/2 + c(z_i)/2.$$

Hence, $c(x_i) + c(y_i) + c(z_i) = B$, for $i = 1, 2, \dots, m$, and the partition $L_1, L_2, \dots, L_m$ also provides a yes answer to (P2).

Conversely, if a partition $L_1, L_2, \dots, L_m$ provides a yes answer to (P2), it follows directly from the definition of $s_i(a)$, for $i = 1, 2, 3$ and $a \in X \cup Y \cup Z$ that this partition also provides a yes answer to (P1) with $n = 3m$ and $K = m$.

□

From Lemma 3, we can immediately conclude that MDVPP with at least one coordinate larger that 1/2 is NP-hard for any $m \geq 3$.

References

[1] Garey, M.R., Graham, R.L., Johnson, D.S.: Resource Constrained Scheduling as Generalized Bin Packing, *J. Combinatorial Theory (A)*, 21(1976), 257-298.

[2] Garey, M.R., Johnson, D.S.: *Computers and Intractability: a Guide to the theory of NP-Completeness*, Freeman, New York, 1979.

[3] Papadimitriu, C.H., Steiglitz, K.: *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, New Jersey, 1982.

(Received April 15,1990)

Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1990

Product hierarchies of automata and homomorphic simulation

P. Dõmõsi [*] Z. Ésik [†‡]

Abstract

A ν_i-product is a network of automata such that each automaton is fed back to at most i of the component automata. We show that the ν_i-hierachy is proper with respect to homomorphic simulation.

For all notions and notations not defined here, see [2], [3] or [6]. An *automaton* $\mathbf{A} = (A, X, \delta)$ is a finite automaton with *state set* A, *input set* X and *transition* $\delta : A \times X \to A$. The transition is also used in the extended sense, i.e. as a function $\delta : A \times X^* \to A$ where X^* is the free monoid of all words over X.

Let $\mathbf{A} = \mathbf{A}_1 \times \ldots \times \mathbf{A}_n(X, \varphi)$ be a *general product* (or *g-product*) of automata $\mathbf{A}_j = (A_j, X_j, \delta_j)$, $j = 1, \ldots, n$, $n \geq 1$. A function

$$\gamma : \{1, \ldots, n\} \to 2^{\{1, \ldots, n\}}$$

is a *neighbourhood function* of $\mathbf{A}$ if each feedback function φ_j is independent of the actual state of any component $\mathbf{A}_k$ with $k \notin \gamma(j)$. Thus the concept of a general product with a neighbourhood function is essentially the same as the automata networks of [7]. A general product $\mathbf{A}$ with a neighbourhood function satisfying $\mathrm{card}(\gamma(j)) \leq i$ for all $j = 1, \ldots, n$, where i is a fixed positive integer, is referred to a *ν_i-product*, cf. [4]. An *$\alpha_0 - \nu_i$-product* is a ν_i-product which is also an α_0-product (i.e. *loop-free product*).

Let $\mathbf{A} = (A, X, \delta)$ and $\mathbf{B} = (B, Y, \delta')$ be automata. We say that $\mathbf{A}$ *homomorphically simulates* $\mathbf{B}$ if there are $A' \subseteq A$ and mappings $h_1 : A' \to B$ and $h_2 : Y \to X^*$ such that h_1 is onto, moreover, $\delta(a, h_2(y)) \in A'$ and

$$h_1(\delta(a, h_2(y))) = \delta'(h_1(a), y)$$

for all $a \in A'$ and $y \in Y$. The function h_2 will be used also in the extended sense, i.e. as a monoid homomorphism $Y^* \to X^*$. Thus $\mathbf{A}$ homomorphically simulates $\mathbf{B}$ if and only if the transformation monoid corresponding to $\mathbf{B}$ is *covered* by the transformation monoid corresponding to $\mathbf{A}$, cf. [5]. If $X = Y$ and $\mathbf{B}$ is a homomorphic image of a subautomaton of $\mathbf{A}$ then $\mathbf{B}$ is *homomorhpically realized* by $\mathbf{A}$, cf. [6].

[*]L. Kossuth University, Mathematical Institute, Debrecen, Egyetem tér 1, H-4032

[†]A. József University, Bolyai Institute, Szeged, Aradi vértanúk tere 1, H-6720

[‡]This research of the second author was carried out with the assistance of the Alexander von Humboldt Foundation.

Let K be a class of automata and let β refer to one of the above particular cases of the g-product. If an automaton $\mathbf{A}$ is homomorphically realized (simulated) by a β-product of automata from K then we write $\mathbf{A} \in HSP_\beta(K)$ ($\mathbf{A} \in HS^*P_\beta(K)$).

Now let $n \geq 1$ be an integer and let $\mathbf{C}_n = (C_n, \{x\}, \delta_n)$ with $C_n = \{0, \ldots, n-1\}$ and $\delta_n(i, x) = i + 1 \bmod n$, for all $i \in C_n$. Thus $\mathbf{C}_n$ is a *counter* with length n. Let $\mathbf{E} = (E, \{x, y\}, \delta_0)$ be an *elevator,* so that $E = \{0, 1\}$, $\delta_0(0, x) = 0$ and $\delta_0(0, y) = \delta_0(1, x) = \delta_0(1, y) = 1$. We set

$$K = \{\mathbf{E}\} \cup \{\mathbf{C}_p | p > 1 \text{ is a prime}\}$$

and prove that there exists an automaton $\mathbf{M} \in HSP_{\alpha_0-\nu_{i+1}}(K)$ which does not belong to $HS^*P_{\nu_i}(K)$, where $i \geq 1$ is any fixed integer.

Let m be the product of the first $i+1$ prime numbers. We define $\mathbf{M} = (M, \{x, y\}, \delta')$ with $M = \{0, \ldots, m\}$ and

$$\delta'(j, x) = \begin{cases} j + 1 \bmod m & \text{if } j = 0, \ldots, m-1 \\ m & \text{if } j = m \end{cases}$$

$$\delta'(j, y) = \begin{cases} j + 1 \bmod m & \text{if } j = 1, \ldots, m-1 \\ m & \text{if } j = 0 \text{ or } j = m. \end{cases}$$

Proof that $\mathbf{M} \notin HS^*P_{\nu_i}(K)$. Assume to the contrary that a ν_i-product with neighbourhood function γ

$$\mathbf{A} = (A, X, \delta) = \mathbf{A}_1 \times \ldots \times \mathbf{A}_n(X, \varphi)$$

of automata form K homomorphically simulates $\mathbf{M}$. We may suppose that n is minimal with this property, i.e., if $\mathbf{B}$ is a ν_i-product of automata from K which homomorphically simulates $\mathbf{M}$, then the number of factors of $\mathbf{B}$ is at least n. Let $A' \subseteq A$ and let $h_1 : A' \to M$, $h_2 : \{x, y\} \to X^*$ be mappings such that h_1 is onto and

$$\delta'(h_1(a), z) = h_1(\delta(a, h_2(z)))$$

for all $a \in A'$ and $z = x, y$, where it is assumed that $\delta(a, h_2(z)) \in A'$. We may choose A' and the functions h_1 and h_2 such that card(A') is minimal.

Let us partition A' as $A' = A_0 \cup A_1$ where $A_0 = h_1^{-1}(M - \{m\})$ and $A_1 = h_1^{-1}(m)$. If $a \in A_0$ and $b \in A'$ then, by the minimality of card(A'), there is a word $u \in \{x, y\}^*$ with $\delta(a, h_2(u)) = b$. Therefore, if $pr_j(a_0) = 1$ and $\mathbf{A}_j = \mathbf{E}$ for some $j = 1, \ldots, n$ and $a_0 \in A_0$, then $pr_j(a) = 1$ for all $a \in A'$. (Of course, pr_j denotes the j-th projection.) But then we can get rid of the j-th component obtaining a ν_i-product of $n-1$ factors that homomorphically simulates $\mathbf{M}$. Since this contradicts the minimality of n we have $pr_j(a) = 0$ for all $a \in A_0$ and $j \in \{1, \ldots, n\}$ with $\mathbf{A}_j = E$. By the construction of $\mathbf{A}$ and the minimality of card(A') it is easy to see that for every $a \in A_1$ there exists $j \in \{1, \ldots, n\}$ with $pr_j(a) = 1$ and $\mathbf{A}_j = \mathbf{E}$.

Now let $a \in h_1^{-1}(0)$ be a fixed state. We have $\delta(a, h_2(y)) \in A_1$, so that $pr_j(\delta(a, h_2(y))) = 1$ and $\mathbf{A}_j = \mathbf{E}$ for some $j \in \{1, \ldots, n\}$. Let $\gamma(j) = \{j_1, \ldots j_t\}$, $t \leq i$. For $s = 1, \ldots, t$, define $r_s = p$ if $\mathbf{A}_j = \mathbf{C}_p$ and $r_s = 1$ if $\mathbf{A}_j = \mathbf{E}$. Let r be the product of the integers r_s. It is clear that m is not a divisor of r. Thus, for $u = h_2(x)$, $\delta(a, u^r) = b \in h_1^{-1}(q)$ with $q \in \{1, \ldots, m-1\}$. Since $pr_{j_s}(b) = pr_{j_s}(a)$ for all $s = 1, \ldots, t$, it follows that $pr_j(\delta(b, h_2(y))) = 1$, which contradicts $\delta(b, h_2(y)) \in A_0$.

Proof that $\mathbf{M} \in HSP_{\alpha_0-\nu_{i+1}}(K)$. For each $j = 1, \ldots, i+1$, let p_j denote the j-th prime number. We construct an $\alpha_0 - \nu_{i+1}$-product

$$\mathbf{A} = \mathbf{C}_{p_1} \times \ldots \times \mathbf{C}_{p_{i+1}} \times \mathbf{E}(\{x, y\}, \varphi)$$

with

$$\varphi_j(k_1, \ldots k_{i+1}, k, z) = \begin{cases} y & \text{if } k_1 = \ldots = k_{i+1} = 0, j = i+2 \text{ and } z = y \\ x & \text{otherwise.} \end{cases}$$

It is straightforward to show that **A** maps homomorphically onto **M**.

Theorem 1 *The ν_i-hierarchy is proper with respect to both homomorphic simulation and homomorphic realization. There exists a class K with the following properties, where $i \geq 1$ is any integer:*

(i) $HSP_{\nu_i}(K) \subset HSP_{\nu_{i+1}}(K)$,

(ii) $HS^*P_{\nu_i}(K) \subset HS^*P_{\nu_{i+1}}(K)$,

(iii) $HSP_{\alpha_0-\nu_i}(K) \subset HSP_{\alpha_0-\nu_{i+1}}(K)$,

(iv) $HS^*P_{\alpha_0-\nu_i}(K) \subset HS^*P_{\alpha_0-\nu_{i+1}}(K)$.

Remarks. For the class K exhibited in the proof we even have $HSP_{\alpha_0}(K) = HSP_g(K)$. Consequently

$$HSP_{\nu_i}(K) \subset HSP_{\alpha_0}(K) \text{ and } HS^*P_{\nu_i}(K) \subset HS^*P_{\alpha_0}(K)$$

hold, too. One might wish to modify the definition of homomorphic simulation by requiring that only nonempty words occur in the range of function h_2. Our result holds with the same proof for this notion or homomorphic simulation, too. Part i) has been already proved in [2] and part ii) in [1]. Nevertheless the class K given above is considerably simpler than that in [1] or [2].

References

[1] Dömösi, P., Products of automata and homomorphic simulation, Papers on Automata and Languages, K. Marx Univ. of Economics, Dept. of Math., Budapest, submitted.

[2] Dömösi, P., and Ésik, Z., On the hierarchy of ν_i-products of automata, Acta Cybernet., 8(1988), 253-257.

[3] Dömösi, P., and Ésik, Z., On homomorphic simulation of automata by α_0-products, Acta Cybernet., 8(1988), 315-323.

[4] Dömösi, P. and Imreh, B., On ν_i-products of automata, Acta Cybernet., 6(1983), pp. 149-162.

[5] Eilenberg, S., Automata, Languages and Machines, vol. B, Academic Press, New York, 1976.

[6] Gécseg, F., Products of Automata, Springer-Verlag, Berlin, 1986.

[7] Tchuente, M., Computation on finite networks of automata, in: C. Choffrut (Ed.), Automata Networks, LNCS 316, Springer-Verlag, Berlin, 1986, 53-67.

(Received June 16, 1989)

Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1990

A note on the axiomatization of iteration theories

Z. Ésik*

Institut für Informatik
Technische Universität München
Arcisstr. 21, D-8000 München 2 Deutschland

Abstract

Iteration theories are a basic underlying structure in many investigations in theoretical computer science. The paper contains some remarks on the aximatization of iteration theories.

Iteration theories were defined in [BEW1] and [BEW2] as the variety generated by pointed iterative theories, which are iterative theories with the operation of iteration made totally defined in an essentially unique way, cf. [E]. Evidence gathered since that time indicates that iteration theories are the basic underlying stucture whenever one is interested in solving fixed point equations (see [BÉs2], [BÉs3], [BÉs4], [BÉs5], [BÉsT] and [St2] for some recent results). One axiomatization of iteration theories was given in [És1]. The purpose of the present note is to present a "scalar axiomtization", one which involves as much as possible morphisms $1 \rightarrow p$. An application of this axiomatization appears in [BÉs5]. We assume the reader is familiar with algebraic theories as defined e.g. in [E], [BÉs1], or [És1]. A preiteration theory is an algebraic theory with an operation of iteration subject to no particular condition. Recall that iteration maps a morphism $f : n \rightarrow n + p$ to $f^\dagger : n \rightarrow p$.

1. Theorem [És1] A preiteration theory is an iteration theory if and only if it satisfies the following identities.
1.1. *Left zero identity*

$$(0_n \oplus f)^\dagger = f, \quad f : n \rightarrow p$$

*Permanent address: A. József University, Bolyai Institute, Szeged, Hungary. Research supported by the Alexander von Humboldt Foundation.

1.2. *Right zero identity*

$$(f \oplus 0_q)^\dagger = f^\dagger \oplus 0_q, \qquad f : n \to n+p$$

1.3. *Dual pairing identity*

$$\langle f, g\rangle^\dagger = \langle h^\dagger, (g \cdot (\rho \oplus \mathbf{1}_p))^\dagger \cdot \langle h^\dagger, \mathbf{1}_p\rangle\rangle, \qquad f : n \to n+m+p, \quad g : m \to n+m+p$$

where $\rho = \langle 0_m \oplus \mathbf{1}_n, \mathbf{1}_n \oplus 0_m\rangle$ is the block transposition $n+m \to m+n$ and

$$h = f \cdot \langle \mathbf{1}_n \oplus 0_p, (g \cdot (\rho \oplus \mathbf{1}_p))^\dagger, 0_n \oplus \mathbf{1}_p\rangle : n \to n+p$$

1.4. *Commutative identity*

$$\langle \mathbf{1}_m \cdot \rho \cdot f \cdot (\rho_1 \oplus \mathbf{1}_p), \ldots, m_m \cdot \rho \cdot f \cdot (\rho_m \oplus \mathbf{1}_p)\rangle^\dagger = \rho \cdot (f \cdot (\rho \oplus \mathbf{1}_p))^\dagger$$

where $f : n \to m+p$, $\rho : m \to n$ is surjective base, and where each $\rho_i : m \to m$ is base with $\rho_i \cdot \rho = \rho$.

2. The above identities have a number of consequences. In particular, the following identities hold in any iteration theory.
2.1. *Fixed point identity*

$$f \cdot \langle f^\dagger, \mathbf{1}_p\rangle = f^\dagger, \quad f : n \to n+p$$

2.2. *Pairing identity*

$$\langle f, g\rangle^\dagger = \langle f^\dagger \cdot \langle k^\dagger, \mathbf{1}_p\rangle, k^\dagger\rangle, \quad f : n \to n+m+p, \qquad g : m \to n+m+p$$

where $k = g \cdot \langle f^\dagger, \mathbf{1}_{m+p}\rangle$
2.3. *Permutation identity*

$$(\rho \cdot f \cdot (\rho^{-1} \oplus \mathbf{1}_p))^\dagger = \rho \cdot f^\dagger, \quad f : n \to n+p$$

where $\rho : n \to n$ is a base permutation.

3. Remark The permutation identity is a special case of the commutative identity. The fact that the fixed point identity is implied by the conditions 1.1-1.4 was only recognized in [És3]. A stronger statement, following a suggestion of a referee, is proved in Lemma 5 below. For the pairing identity we mention the following result.

4. Lemma Let T be a preiteration theory which satisfies the permutation identity. Then the pairing identity holds in T if and only if the dual pairing identity holds.

Proof. We only prove that the dual pairing identity is implied by the pairing identity and the permutation identity. Let $f : n \to n+m+p$, $g : m \to n+m+p$ and, as before, denote by ρ the block transposition $n+m \to m+n$. By the permutation identity and the pairing identity we have

$$\begin{aligned}
\langle f, g\rangle^\dagger &= (\rho \cdot \rho^{-1} \cdot \langle f, g\rangle \cdot (\rho \oplus \mathbf{1}_p) \cdot (\rho^{-1} \oplus \mathbf{1}_p))^\dagger \\
&= \rho \cdot (\rho^{-1} \cdot \langle f, g\rangle \cdot (\rho \oplus \mathbf{1}_p))^\dagger \\
&= \rho \cdot \langle g \cdot (\rho \oplus \mathbf{1}_p), f \cdot (\rho \oplus \mathbf{1}_p)\rangle^\dagger \\
&= \rho \cdot \langle (g \cdot (\rho \oplus \mathbf{1}_p))^\dagger \cdot \langle k^\dagger, \mathbf{1}_p\rangle, k^\dagger\rangle \\
&= \langle k^\dagger, (g \cdot (\rho \oplus \mathbf{1}_p))^\dagger \cdot \langle k^\dagger, \mathbf{1}_p\rangle\rangle,
\end{aligned}$$

where

$$\begin{aligned} k &= f \cdot (\rho \oplus \mathbf{1}_p) \cdot \langle (g \cdot (\rho \oplus \mathbf{1}_p))^\dagger, \mathbf{1}_{n+p} \rangle \\ &= f \cdot \langle \mathbf{1}_n \oplus 0_p, (g \cdot (\rho \oplus \mathbf{1}_p))^\dagger, 0_n \oplus \mathbf{1}_p \rangle. \end{aligned}$$

The proof is complete.

Besides that presented in Theorem 1, several equivalent axiomatizations of iteration theories are known. The following results are taken from [St1],[CSt] and [És2].

5. Lemma [CSt] Let T be a preiteration theory which satisfies the left zero identity 1.1, the dual pairing identity 1.3 and the permutation identity 2.3. Then the fixed point identity 2.1 holds in T.

Proof. Let $f : n \to n + p$ and define $g = \langle 0_n \oplus f, \mathbf{1}_n \oplus 0_{n+p} \rangle$. Using 1.1 and 1.3 we obtain

$$\begin{aligned} (\mathbf{1}_n \oplus 0_n) \cdot g^\dagger &= ((0_n \oplus f) \cdot \langle \mathbf{1}_n \oplus 0_p, (0_n \oplus \mathbf{1}_n \oplus 0_p)^\dagger, 0_n \oplus \mathbf{1}_p \rangle)^\dagger \\ &= ((0_n \oplus f) \cdot \langle \mathbf{1}_n \oplus 0_p, \mathbf{1}_n \oplus 0_p, 0_n \oplus \mathbf{1}_p \rangle)^\dagger \\ &= f^\dagger. \end{aligned}$$

Similarly, by 1.1 and 2.2,

$$\begin{aligned} (\mathbf{1}_n \oplus 0_n) \cdot g^\dagger &= (0_n \oplus f)^\dagger \cdot \langle ((\mathbf{1}_n \oplus 0_{n+p}) \cdot \langle (0_n \oplus f)^\dagger, \mathbf{1}_{n+p} \rangle)^\dagger, \mathbf{1}_p \rangle \\ &= f \cdot \langle ((\mathbf{1}_n \oplus 0_{n+p}) \cdot \langle f, \mathbf{1}_{n+p} \rangle)^\dagger, \mathbf{1}_p \rangle \\ &= f \cdot \langle f^\dagger, \mathbf{1}_p \rangle. \end{aligned}$$

The proof is completed using Lemma 4.

6. Theorem [St1], [CSt] A preiteration theory is an iteration theory if and only if the following hold.

6.1. *Parameter identity*

$$(f \cdot (\mathbf{1}_n \oplus g))^\dagger = f^\dagger \cdot g, \qquad f : n \to n + p,\ g : p \to q$$

6.2. *Composition identity*

$$f \cdot \langle (g \cdot \langle f, 0_m \oplus \mathbf{1}_p \rangle)^\dagger, \mathbf{1}_p \rangle = (f \cdot \langle g, 0_n \oplus \mathbf{1}_p \rangle)^\dagger, \qquad f : n \to m + p,\ g : m \to n + p$$

6.3 *Double dagger identity*

$$(f \cdot (\langle \mathbf{1}_n, \mathbf{1}_n \rangle \oplus \mathbf{1}_p))^\dagger = f^{\dagger\dagger}, \qquad f : n \to n + n + p$$

6.4. The commutative identity 1.4.

7. Theorem [És2] A preiteration theory is an iteration theory if and only if the following identities hold.

7.1. The *special left zero identity*, i.e. 1.1 with $n = 1$.
7.2. The *special parameter identity*, i.e. 6.1 with $n = 1$.
7.3. The *special dual pairing identity* 1.3 with $m = 1$.
7.4. *Special permutation identity*

$$f^\dagger \cdot \langle (g \cdot \langle f^\dagger, \mathbf{1}_{1+p}\rangle)^\dagger, \mathbf{1}_p\rangle = (f \cdot \langle \mathbf{1}_1 \oplus 0_p, (g \cdot (\rho \oplus \mathbf{1}_p))^\dagger, 0_1 \oplus \mathbf{1}_p\rangle)^\dagger$$

where $f, g : 1 \to 1 + 1 + p$ and $\rho : 2 \to 2 + p$ is the nontrivial base permutation.
7.5. *Special commutative identity*

$$\mathbf{1}_m \cdot \langle 1_m \cdot \rho \cdot f \cdot (\rho_1 \oplus \mathbf{1}_p), \ldots, m_m \cdot \rho \cdot f \cdot (\rho_m \oplus \mathbf{1}_p)\rangle^\dagger = 1_n \cdot (f \cdot (\rho \oplus \mathbf{1}_p))^\dagger$$

where $f : n \to m + p$, $\rho : m \to n$ is a monotone surjective base morphism, and where each $\rho_i : m \to m$ is base with $\rho_i \cdot \rho = \rho$.

8. Remark In addition to 7.1-7.5, the *special fixed point identity* 2.1 with $n = 1$ was also required in [És2]. This is however already implied, for taking $f = 0_1 \oplus h$ and $g = 1_{2+p} = \mathbf{1}_1 \oplus 0_{1+p}$ in the special permutation identity we obtain

$$\begin{aligned} f^\dagger \cdot \langle (g \cdot \langle f^\dagger, \mathbf{1}_{1+p}\rangle)^\dagger, \mathbf{1}_p\rangle &= (0_1 \oplus h)^\dagger \cdot \langle (1_{2+p} \cdot \langle (0_1 \oplus h)^\dagger, \mathbf{1}_{1+p}\rangle)^\dagger, \mathbf{1}_p\rangle \\ &= h \cdot \langle h^\dagger, \mathbf{1}_p\rangle \end{aligned}$$

and

$$\begin{aligned} &(f \cdot \langle \mathbf{1}_1 \oplus 0_p, (g \cdot (\rho \oplus \mathbf{1}_p))^\dagger, 0_1 \oplus \mathbf{1}_p\rangle)^\dagger = \\ &= ((0_1 \oplus h) \cdot \langle \mathbf{1}_1 \oplus 0_p, (0_1 \oplus \mathbf{1}_1 \oplus 0_p)^\dagger, 0_1 \oplus \mathbf{1}_p\rangle)^\dagger \\ &= ((0_1 \oplus h) \cdot \langle \mathbf{1}_1 \oplus 0_p, \mathbf{1}_1 \oplus 0_p, 0_1 \oplus \mathbf{1}_p\rangle)^\dagger \\ &= h^\dagger \end{aligned}$$

by 7.1. Thus $h \cdot \langle h^\dagger, \mathbf{1}_p\rangle = h^\dagger$.

We note that it is enough to require the special parameter identity $(f \cdot (\mathbf{1}_1 \oplus g))^\dagger = f^\dagger \cdot g$ in Theorem 7 only for $f : 1 \to 1 + p$ and base $g : p \to q$, cf. [És2]. A result closely related to Theorem 7 was found independently in [CU], see also [Ca]. As a part of the proof of Theorem 7, the following result was established in [És2]. A scalar preiteration theory is an algebraic theory with iteration defined on scalar morphisms $f : 1 \to 1 + p$.

9. Theorem [És2] Let T be a scalar preiteration theory satisfying the special parameter identity and the special permutation identity. Extend the definition of iteration by the special dual pairing identity, i.e. let $0^\dagger_{1+p} = 0_p$ and for $f : n \to n + 1 + p$ and $g : 1 \to n + 1 + p$,

$$\langle f, g\rangle^\dagger = \langle h^\dagger, (g \cdot (\rho \oplus \mathbf{1}_p))^\dagger \cdot \langle h^\dagger, \mathbf{1}_p\rangle\rangle$$

with ρ and h as in 1.3. Then T becomes a preiteration theory in which the identities 1.1-1.3, 2.1 and 2.3 hold.

10. Corollary The preiteration theory of Theorem 9 also satisfies the parameter identity 6.1, the composition identity 6.2 and the double dagger identity 6.3.

Proof. It is proved in [Ésl] that any preiteration theory T in which 1.1-1.3 and 2.3 hold satisfies the parameter identity. By the main result of [BÉsl], T satisfies any identity valid for flowchart schemes. Therefore the double dagger identity holds in T. A direct proof, starting with $a = \langle f, \mathbf{1}_n \oplus 0_{n+p}\rangle$, $f : n \to n+n+p$, may be obtained by a calculation similar to that given below (cf. [Stl]). Now for the composition identity. Given $f : n \to m+p$ and $g : m \to n+p$, define

$$a = \langle 0_n \oplus f, g \cdot (\mathbf{1}_n \oplus 0_m \oplus \mathbf{1}_p)\rangle : n+m \to n+m+p.$$

The pairing identity 2.2, which holds in T by Lemma 4, and the left zero identity 1.1 imply

$$\begin{aligned}(\mathbf{1}_n \oplus 0_m) \cdot a^\dagger &= (0_n \oplus f)^\dagger \cdot \langle (g \cdot (\mathbf{1}_n \oplus 0_m \oplus \mathbf{1}_p) \cdot \langle (0_n \oplus f)^\dagger, \mathbf{1}_{m+p}\rangle)^\dagger, \mathbf{1}_p\rangle \\ &= f \cdot \langle (g \cdot (\mathbf{1}_n \oplus 0_m \oplus \mathbf{1}_p) \cdot \langle f, \mathbf{1}_{m+p}\rangle)^\dagger, \mathbf{1}_p\rangle \\ &= f \cdot \langle (g \cdot \langle f, 0_m \oplus \mathbf{1}_p\rangle)^\dagger, \mathbf{1}_p\rangle.\end{aligned}$$

By the dual pairing identity 1.3 and the left zero identity again,

$$\begin{aligned}&(\mathbf{1}_n \oplus 0_m) \cdot a^\dagger = \\ &= ((0_n \oplus f) \cdot \langle \mathbf{1}_n \oplus 0_p, (g \cdot (\mathbf{1}_n \oplus 0_m \oplus \mathbf{1}_p) \cdot (\rho \oplus \mathbf{1}_p))^\dagger, 0_n \oplus \mathbf{1}_p\rangle)^\dagger \\ &= ((0_n \oplus f) \cdot \langle \mathbf{1}_n \oplus 0_p, (0_m \oplus g)^\dagger, 0_n \oplus \mathbf{1}_p\rangle)^\dagger \\ &= (f \cdot \langle g, 0_n \oplus \mathbf{1}_p\rangle)^\dagger,\end{aligned}$$

where ρ denotes the block transposition $n+m \to m+n$.

Except for 7.5, the axiomatization given in Theorem 7 is based on scalar iteration, for the special dual pairing identity can be thought of as a definition of vector iteration (together with $0_p^\dagger = 0_p$ which is already forced by the theory identities). Nevertheless both sides of the special permutation identity can be expressed in terms of scalar iteration. Below we present another axiomatization of this sort.

11. Theorem Let T be a scalar preiteration theory such that the *special parameter identity*, the *special composition identity* 6.2 with $n = m = 1$, and the *special double dagger identity* 6.3 with $n = 1$ hold. If iteration is extended by the special dual pairing identity then T becomes a preiteration theory satisfying the identities 1.1 - 1.3, 2.1 and 2.3. Moreover, T satisfies the parameter identity, the composition identity and the double dagger identity.

Proof. We show that the special fixed point identity and the special permutation identity hold in T. For the special fixed point identity just take $g = \mathbf{1}_1 \oplus 0_p = 1_{1+p}$ in the special composition identity

$$f \cdot \langle (g \cdot \langle f, 0_1 \oplus \mathbf{1}_p\rangle)^\dagger, \mathbf{1}_p\rangle = (f \cdot \langle g, 0_1 \oplus \mathbf{1}_p\rangle)^\dagger.$$

For the special permutation identity first we prove that

$$(g\cdot\langle f^\dagger,\mathbf{1}_{1+p}\rangle)^\dagger=((g\cdot(\rho\oplus\mathbf{1}_p))^\dagger\cdot\langle f^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger, \tag{11.1}$$

where $\rho:2\to2$ is the nontrivial base permutation. We use the special parameter identity and the special double dagger identity.

$$\begin{aligned}
&((g\cdot(\rho\oplus\mathbf{1}_p))^\dagger\cdot\langle f^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger=\\
&=\ (g\cdot(\rho\oplus\mathbf{1}_p)\cdot(\mathbf{1}_1\oplus\langle f^\dagger,0_1\oplus\mathbf{1}_p\rangle))^{\dagger\dagger}\\
&=\ (g\cdot(\rho\oplus\mathbf{1}_p)\cdot\langle\mathbf{1}_1\oplus0_{1+p},0_1\oplus f^\dagger,0_2\oplus\mathbf{1}_p\rangle)^{\dagger\dagger}\\
&=\ (g\cdot\langle0_1\oplus f^\dagger,\mathbf{1}_1\oplus0_{1+p},0_2\oplus\mathbf{1}_p\rangle)^{\dagger\dagger}\\
&=\ (g\cdot\langle0_1\oplus f^\dagger,\mathbf{1}_1\oplus0_{1+p},0_2\oplus\mathbf{1}_p\rangle\cdot(\langle\mathbf{1}_1,\mathbf{1}_1\rangle\oplus\mathbf{1}_p))^\dagger\\
&=\ (g\cdot\langle0_1\oplus f^\dagger,\mathbf{1}_1\oplus0_{1+p},0_2\oplus\mathbf{1}_p\rangle\cdot\langle\mathbf{1}_1\oplus0_p,\mathbf{1}_1\oplus0_p,0_1\oplus\mathbf{1}_p\rangle)^\dagger\\
&=\ (g\cdot\langle f^\dagger,\mathbf{1}_{1+p}\rangle)^\dagger
\end{aligned}$$

Next, by the special composition identity,

$$f^\dagger\cdot\langle((g\cdot(\rho\oplus\mathbf{1}_p))^\dagger\cdot\langle f^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger,\mathbf{1}_p\rangle=(f^\dagger\cdot\langle(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger. \tag{11.2}$$

Finally, we observe that

$$(f\cdot\langle\mathbf{1}_1\oplus0_p,(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger=(f^\dagger\cdot\langle(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger. \tag{11.3}$$

Indeed, by the special parameter identity and the special double dagger identity,

$$\begin{aligned}
&(f^\dagger\cdot\langle(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger=\\
&=\ (f\cdot(\mathbf{1}_1\oplus\langle(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_1\oplus\mathbf{1}_p\rangle))^{\dagger\dagger}\\
&=\ (f\cdot\langle\mathbf{1}_1\oplus0_{1+p},0_1\oplus(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_2\oplus\mathbf{1}_p\rangle)^{\dagger\dagger}\\
&=\ (f\cdot\langle\mathbf{1}_1\oplus0_{1+p},0_1\oplus(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_2\oplus\mathbf{1}_p\rangle\cdot(\langle\mathbf{1}_1,\mathbf{1}_1\rangle\oplus\mathbf{1}_p))^\dagger\\
&=\ (f\cdot\langle\mathbf{1}_1\oplus0_p,(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger.
\end{aligned}$$

The proof is now easily completed. By 11.1-11.3,

$$\begin{aligned}
(f\cdot\langle\mathbf{1}_1\oplus0_p,(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger&=(f^\dagger\cdot\langle(g\cdot(\rho\oplus\mathbf{1}_p))^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger\\
&=f^\dagger\cdot\langle((g\cdot(\rho\oplus\mathbf{1}_p))^\dagger\cdot\langle f^\dagger,0_1\oplus\mathbf{1}_p\rangle)^\dagger,\mathbf{1}_p\rangle\\
&=f^\dagger\cdot\langle(g\cdot\langle f^\dagger,\mathbf{1}_{1+p}\rangle)^\dagger,\mathbf{1}_p\rangle.
\end{aligned}$$

Theorem 11 implies Theorem 2, Chapter 13 in [C] for matrix theories, see also [BÉs5]. A consequence of Theorem 11 and the previous results is given below.

12. Theorem A preiteration theory is an iteration theory if and only if the special parameter identity, the special composition identity, the special double dagger identity, the special dual pairing identity and the special commutative identity hold.

It is interesting to compare Theorem 12 with the following result, essentially taken from [És2].

13. Theorem [És2] A preiteration theory is an iteration theory if and only if the special left zero identity, the special parameter identity, the special pairing identity and the following variant of the commutative identity hold:

$$1_m\cdot\langle 1_m\cdot\rho\cdot f\cdot(\rho_1\oplus \mathbf{1}_p),\ldots,m_m\cdot\rho\cdot f\cdot(\rho_m\oplus\mathbf{1}_p)\rangle^\dagger = 1_m\cdot\rho\cdot(f\cdot(\rho\oplus\mathbf{1}_p))^\dagger \qquad 13.1$$

where f,ρ and $\rho_i, i=1,\ldots,m$, are as in the commutative identity 1.4.

14. Remark By Lemma 4, the dual pairing identity 1.3 can be replaced by the pairing identity 2.2 in Theorem 1. Similarly, we may use the special pairig identity 2.2 with $m=1$ (or $n=1$) in Theorems 7, 9, 12 and 13. In Theorems 7, 9, 11 and 13 one can also use the special symmetric pairing identity

$$\langle f,g\rangle^\dagger = \langle h^\dagger, k^\dagger\rangle,$$

where $f:n\to n+1+p$, $g:1\to n+1+p$, and where h and k are defined as in 1.3 and 2.2. In Theorem 12, instead of the special commutative identity, we may require 13.1 for monotone surjective $\rho:m\to n$.

Let T be a preiteration theory such that the the parameter identity, the permutation identity and the dual pairing identity (or pairing identity) hold. Suppose that

$$f\cdot(\alpha\oplus\mathbf{1}_p)=\alpha\cdot g$$

for $f:n\to n+p$, $g:m\to m+p$ and an injective base morphism $\alpha:n\to m$. It is a routine calculation to prove that

$$f^\dagger=\alpha\cdot g^\dagger.$$

A preiteration theory has a *weak functorial dagger* if

$$f\cdot(\rho\oplus\mathbf{1}_p)=\rho\cdot g\Rightarrow f^\dagger=\rho\cdot g^\dagger \qquad 14.1$$

for all $f:n\to n+p$, $g:m\to m+p$ and surjective base morphism $\rho:n\to m$. It is known that the commutative identity holds in any preiteration theory with weak functorial dagger, cf. [És1]. Most known iteration theories have weak functorial dagger. The existence of an iteration theory not satisfying 14.1 was pointed out in [És4].

15. Proposition Let T be a preiteration theory such that the parameter identity, the permutation identity and the dual pairing (or pairing) identity hold. Then T has weak functorial dagger if and only if 14.1 holds with $m=1$.

Proof. Since the permutation identity holds in T, it suffices to prove 14.1 for monotone surjective base morphisms $\rho:n\to m$. Our argument uses induction on m. The basis case $m=1$ holds by assumption. Supposing the statement holds for $m\geq 1$, let $f:n\to n+p$, $g:m+1\to m+1+p$ and $\rho:n\to m+1$ be such that

$$f\cdot(\rho\oplus\mathbf{1}_p)=\rho\cdot g, \qquad 15.1$$

where $\rho : n \to m+1$ is a monotone surjective base morphism. We can write

$$\begin{array}{lll} f = & \langle f_1, f_2\rangle, & f_1 : n_1 \to n+p,\ f_2 : n_2 \to n+p \\ g = & \langle g_1, g_2\rangle, & g_1 : m \to m+1+p,\ g_2 : 1 \to m+1+p \\ \rho = & \rho_1 \oplus \rho_2, & \rho_1 : n_1 \to m,\ \rho_2 : n_2 \to 1 \end{array}$$

where ρ_1 and ρ_2 are monotone surjective base morphisms with

$$f_i \cdot (\rho_1 \oplus \rho_2 \oplus \mathbf{1}_p) = \rho_i \cdot g_i, \quad i = 1, 2. \qquad 15.2$$

The induction hypothesis and the parameter identity yield

$$f_1^\dagger \cdot (\rho_2 \oplus \mathbf{1}_p) = \rho_1 \cdot g_1^\dagger. \qquad 15.3$$

Now let $h = f_2 \cdot \langle f_1^\dagger, \mathbf{1}_{n_2+p}\rangle : n_2 \to n_2 + p$ and $k = g_2 \cdot \langle g_1^\dagger, \mathbf{1}_{1+p}\rangle : 1 \to 1+p$. We have

$$h \cdot (\rho_2 \oplus \mathbf{1}_p) = \rho_2 \cdot k. \qquad 15.4$$

Indeed,

$$\begin{aligned} h \cdot (\rho_2 \oplus \mathbf{1}_p) &= f_2 \cdot \langle f_1^\dagger, \mathbf{1}_{n_2+p}\rangle \cdot (\rho_2 \oplus \mathbf{1}_p) \\ &= f_2 \cdot \langle f_1^\dagger \cdot (\rho_2 \oplus \mathbf{1}_p), \rho_2 \oplus \mathbf{1}_p\rangle \\ &= f_2 \cdot \langle \rho_1 \cdot g_1^\dagger, \rho_2 \oplus \mathbf{1}_p\rangle, \end{aligned}$$

by 15.3,

$$\begin{aligned} &= f_2 \cdot (\rho_1 \oplus \rho_2 \oplus \mathbf{1}_p) \cdot \langle g_1^\dagger, \mathbf{1}_{1+p}\rangle \\ &= \rho_2 \cdot g_2 \cdot \langle g_1^\dagger, \mathbf{1}_{1+p}\rangle \\ &= \rho_2 \cdot k, \end{aligned}$$

by 15.4. From 15.4, by the induction hypothesis, we obtain

$$h^\dagger = \rho_2 \cdot k^\dagger. \qquad 15.5$$

The proof is completed by using the pairing identity.

$$\begin{aligned} f^\dagger &= \langle f_1^\dagger \cdot \langle h^\dagger, \mathbf{1}_p\rangle, h^\dagger\rangle \\ &= \langle f_1^\dagger \cdot \langle \rho_2 \cdot k^\dagger, \mathbf{1}_p\rangle, \rho_2 \cdot k^\dagger\rangle, \end{aligned}$$

by 15.5,

$$\begin{aligned} &= \langle f_1^\dagger \cdot (\rho_2 \oplus \mathbf{1}_p) \cdot \langle k^\dagger, \mathbf{1}_p\rangle, \rho_2 \cdot k^\dagger\rangle \\ &= \langle \rho_1 \cdot g_1^\dagger \cdot \langle k^\dagger, \mathbf{1}_p\rangle, \rho_2 \cdot k^\dagger\rangle, \end{aligned}$$

by 15.3,

$$\begin{aligned} &= (\rho_1 \oplus \rho_2) \cdot \langle g_1^\dagger \cdot \langle k^\dagger, \mathbf{1}_p \rangle, k^\dagger \rangle \\ &= \rho \cdot g^\dagger. \end{aligned}$$

An equivalent statement is proved independently in [B]. Proposition 15 also appears in [BÉs4].

16. Corollary An iteration theory has weak functional dagger if and only if 16.1 $f \cdot (\rho \oplus \mathbf{1}_p) = \rho \cdot g \Rightarrow 1_n \cdot f^\dagger = g^\dagger$ for all $f : n \to n+p, g : 1 \to 1+p$ with $n \geq 1$, where ρ is the unique base morphisms $n \to 1$.

By an example given in [És4], it is not possible to impose an upper bound on the integer n appearing in 16.1. Nevertheless it is enough to require 16.1 for any infinite set of integers n. Combining Corollary 16 with Theorems 1, 5, 6, or 11, one obtains axiomatizations of the quasivariety of iteration theories with weak functorial dagger studied under the name of strong iteration theories in [St1]. Thus we have e.g. the following statement.

17. Corollary A preiteration theory is an iteration theory with weak functorial dagger if and only if it satisfies 16.1 and the special parameter identity, the special composition identity, the special double dagger identity and the special dual pairing identity.

Finally we mention some simplifications of the commutative identity. It is implicit in Lemmas 1.1 and 3.2 in [És1] that the commutative identity reduces to the special case that each ρ_i is a bijective base morphism or that each one is an aperiodic base morphism. Similarly, it suffices to require the special commutative identity in Theorems 7 and 12 above in one of these two cases. However it is not known if it is enough to require the commutative identity for $n = 1$.

Open problem Find an essential simplification of the commutative identity.

References

[B] Bartha, M., On two constuctions of models for systolic sytems, Technical Report, Oxford Univ. Computing Lab., 1988.

[BÉs1] Bloom, S.L. and Ésik, Z., Axiomatizing schemes and their behaviours, J. of Comput. Sys. Sci., 31(1985), 375-393.

[BÉs2] Bloom, S.L. and Ésik, Z., The equational logic of circular data type specifications, Theoret. Comput. Sci., 63(1989), 303-331.

[BÉs3] Bloom, S.L. and Ésik, Z., Floyd-Hoare logic in iteration theories, Research Report, Stevens Institute of Technology, 1988, J. Assoc. Comput. Machinery, to appear.

[BÉs4] Bloom, S.L. and Ésik, Z., Iteration theories: the equational logic of iterative processes, manuscript, 1988.

[BÉs5] Bloom, S.L. and Ésik, Z., Matrix and matricial iteration theories, 1989, submitted for publication.

[BÉsT] Bloom, S.L., Ésik, Z. and Taubner, D., Iteration theories of synchronization trees, Information and Computation, to appear; extended abstract in: Semantics for Concurrency, LNCS, Springer-Verlag, 1990, 96-115.

[BEW1] Bloom, S.L., Elgot, C.C. and Wright, J.B., Solutions of the iteration equation and extensions of the scalar iteration operation, SIAM J. Computing, 9(1980), 26-45.

[BEW2] Bloom, S.L., Elgot, C.C. and Wright, J.B., Vector iteration in pointed iterative theories, SIAM J. Computing, 9(1980), 525-540.

[C] Conway, J.H., Regular algebra and finite machines, Chapman and Hall, London, 1971.

[Ca] Cazanescu, V.E., On algebraic theories with iterate, Rev. Roumaine Math. Pures Appl., 33(1988), 561-573.

[CSt] Cazanescu, V.E. and Stefanescu, Gh., Feedback, iteration and repetition, Research Report 42(1988), National Institute for Scientific and Technical Creation, Bucharest.

[CU] Cazanescu, V.E. and Ungureanu, C., Again an advice on structuring compilers and proving them correct, Preprint Series in Mathematics, 75(1988), National Institute for Scientific and Technical Creation, Bucharest.

[E] Elgot, C.C., Monadic computation and iterative algebraic theories, in: Logic Colloquium'73, Studies in Logic, vol. 80, North Holland, 1975, 175-230.

[És1] Ésik, Z., Identities in iterative and rational algebraic theories, Computational Linguistics and Computer Languages, XIV(1980), 183-207.

[És2] Ésik, Z., On generalized iterative algebraic theories, Computational Linguistics and Computer Languages, XV(1982), 95-110.

[És3] Ésik, Z., Algebras of iteration theories, J. of Comput. Sys. Sci., 27(1983), 291-303.

[És4] Ésik, Z., The independence of the equational axioms of iteration theories, J. of Comput. Sys. Sci., 36(1988), 66-76.

[St1] Stefanescu, Gh., On flowchart theories, Part 1: The deterministic case, J. Comput. Sys. Sci., 35(1986), 163-191.

[St2] Stefanescu, Gh., On flowchart theories, Part 2: The nondeterministic case, Theoret. Comput. Sci., 52(1987), 307-340.

(Received May 19, 1989)

Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1990

Investigations on Armstrong relations, dependency inference, and excluded functional dependencies

G. Gottlob and L. Libkin
Department of Applied Computer Science*
University of Technology
Vienna – Austria

Abstract

This paper first presents some new results on excluded functional dependencies, i.e., FDs which do not hold on a given relation schema. In particular, we show how excluded dependencies relate to Armstrong relations, and we state criteria for deciding whether a set of excluded dependencies characterizes a set of FDs. In the rest of the paper, complexity issues related to the following three problems are studied : to construct an Armstrong relation for a cover F of functional dependencies (FDs), to construct a cover of FDs that hold in a relation R (dependency inference), and, given a cover F and a relation R, to decide if all the FDs that hold in R can be derived from F. The first two problems are known to have exponential complexity. We give a new proof for the second problem by showing that dependency inference can be used to compute all keys of a relation instance. We prove that the third problem is co-$\mathcal{NP}$-complete. Further, it is shown that the problems can be solved in polynomial time if it is known that a relation scheme satisfies some additional properties, which are polynomially recognizable themselves.

1 Introduction

In order to express the information conveyed by a set of functional dependencies (FDs) that hold on a relation scheme, one can alternatively specify the set of all dependencies that *do not* hold on the scheme. These dependencies, called excluded functional dependencies (XFDs), are closely related to Armstrong relations. Note,

*Mailing address: Institut für Angewandte Informatik, TU Wien, Paniglgasse 16, A-1040 Wien, Austria. Internet e-mail of first Author: gottlob@vexpert.at however, that not every arbitrary set of XFDs corresponds to a set of FDs. In this paper we therefore introduce the notion of completeness of sets of XFDs. Informally, a set of XFDs is complete if it unambiguously characterizes a set of FDs. We also present completeness criteria which can be tested in polynomial time.

In the rest of the paper we study complexity issues related to several problems concerning *functional dependencies* (FDs for short) in relational databases. The three problems which we are interested in are the following.

Problem 1 (Constructing Armstrong Relation) [BDFS84], [MR86] *Given a set F of FDs, construct an Armstrong relation R for F.*

Problem 2 (Dependency Inference Problem) [MR87], [MR90] *Given a relation R, construct a cover F of FDs that hold in R.*

Problem 3 (FD-Relation Implication Problem) *Given a relation R and a set F of FDs, decide whether all the FDs that hold in R can be derived from F.*

The first two problems are of high practical importance, see [BDFS84, MR86, MR87, MR89]. However, it is known that these problems are inherently exponential and hence it is impossible to design polynomial algorithms for their solution [BDFS84, MR87, MR86]. The third problem seems to be important for design theory too. To our knowledge, its complexity is still unknown. We show that the problem of finding all the minimal keys of a relation instance can be polynomially transformed to the second problem. Then we prove that the Problem 3 is co-NP-complete.

Let us introduce a new problem which is close to the Problem 3.

Problem 4 (FD-Relation Equivalence Problem) *Given a relation R and a set F of FDs, decide whether the sets of FDs that hold in R and that can be derived from F coincide. In other words: decide whether R is an Armstrong Relation for F.*

This problem can be decomposed into two subproblems:

- *Decide whether all the FDs that hold in R can be derived from F, i.e., whether* $F_R \subseteq F^+$. Note that this subproblem is identical to Problem 3; and
- *Decide whether each FD of F also holds in R, i.e., whether* $F^+ \subseteq F_R$. Note that this subproblem is easily solvable in polynomial time.

Problem 4 thus consists of the conjunction of a co-$\mathcal{NP}$-complete subproblem and a polynomially decidable subproblem. Unfortunately, this knowledge does not allow us to determine its complexity. It seems rather difficult to find the complexity class of Problem 4. To our best knowledge, this problem has never been dealt with in the literature. We therefore want highlight the complexity analysis of Problem 4 as an interesting open problem to which we plan to dedicate further research efforts.

We show that the complexity of Problems 1-4 becomes polynomial if it is known that F satisfies certain additional properties. These additional properties will be formulated for a set F of FDs and for the associated closure operator and semilattice. We also show that these properties can be recognized in polynomial time.

The paper is organized as follows. In Section 2 we state some basic definitions. In Section 3 we derive our new results concerning excluded functional dependencies. In Section 4 we show that the key-generating problem for relation instances can be solved by using dependency inference. The fifth Section is dedicated to the proof of the co-$\mathcal{NP}$-completeness of Problem 3. In Section 6 we study special cases in which our four problems become polynomial. Some concluding remarks are made in Section 7.

2 Basic Definitions

In this section we briefly remind the necessary concepts of relational database theory (cf. [Ma83], [PBGV89]) and state some preliminary results.

Let U be a set of *attributes*. With each attribute $A \in U$ associate its domain $D(A)$. A *relation* (or *relation instance*) over U is a subset of $\prod_{A \in U} D(A)$. We can think of a relation as being a set of tuples $t : U \rightarrow \bigcup_{A \in U} D(A)$ with $t(A) \in D(A)$ for each $A \in U$. Note that some authors distinguish between the terms "relation" and "relation instance" while here both terms have the same meaning.

If X and Y denote sets of attributes and A denotes an attribute, we often write XY, XA, $X - A$, etc. instead of respectively $X \cup Y$, $X \cup \{A\}$, $X - \{A\}$, etc.

A FD is an expression of form $X \rightarrow Y, X, Y \subseteq U$. We say that FD $X \rightarrow Y$ holds in R if for every $t_1, t_2 \in R, t_1(A) = t_2(A)$ for all $A \in X$ implies that $t_1(A) = t_2(A)$ for all $A \in Y$.

The set of all FDs that hold for a given relation R is denoted by F_R. F_R satisfies the following properties : $X \rightarrow Y \in F_R$ for all $Y \subseteq X$(pseudoreflexivity), and $XZ \rightarrow V \in F_R$ if $X \rightarrow Y \in F_R$ and $YZ \rightarrow V \in F_R$ (pseudotransitivity).

If we are given a set F of FDs, F^+ stands for the set of all FDs that can be derived from F by the above rules being used. Of course, for each relation R, $F_R^+ = F_R$. Furthermore, for each set F of functional dependencies, there is a relation R with $F^+ = F_R$; such a relation is called *Armstrong Relation* [FA82].

A set F of FDs is called a *cover* of G if $F^+ = G^+$. A cover F is called *nonredundant* if for each $f \in F$ we have $f \notin (F - f)^+$. A cover F is called *minimum* if $|F| \leq |F'|$ for all other covers F'.

It is well-known that each set F of FDs is equivalent to a set F' of FDs containing only single attributes as right hand sides. Indeed, each FD $X \rightarrow A_1 A_2 \ldots A_n$ can be replaced by the following n FDs: $X \rightarrow A_1$, $X \rightarrow A_2$ $\ldots$, $X \rightarrow A_n$. Therefore, we can always assume without loss of generality that a given set of FDs has only single attributes as right hand sides.

A set X is called a *key* if $X \rightarrow U \in F^+$. A key is called *minimal* if each $Y \subset X$ is not a key.

A pair $< U, F >$ is called a *relation scheme*, or RS for short. A RS is in *Boyce-Codd normal form* (BCNF) if for each $X \to A \in F^+$, where $A \notin X$, it holds: $X \to U \in F^+$.

Given a set F of FDs, define the mapping $C_F(X) = \{A \in U : X \to A \in F^+\}$ (we will write C_R instead of C_{F_R}). Then C_F is a *closure*, that is, $X \subseteq C_F(X), X \subseteq Y$ implies $C_F(X) \subseteq C_F(Y)$ and $C_F(C_F(X)) = C_F(X)$. If F is understood then $C_F(X)$ is also denoted by X^+.

The following well-known algorithm computes the closure $C_F(X)$ of a set of attributes X. Here we assume that F has only single attributes as right hand sides.

```
Algorithm CLOSURE
Input:  a set $F$ of FDs over $U$
        and a set $X \subseteq U$ of attributes.
Output: $C_F(X)$
Method:
result := $X$;
WHILE there exists an attribute $A \in U$ such that
        $A \notin$ result AND
        there is a FD $Y \to A \in F$ such that $Y \subseteq$ result
DO result := result $\cup A$;
RETURN(result).
```

A set X is *closed* (w. r. t. C_F) if $C_F(X) = X$. Denote by S_F the family of all closed sets (again, we write S_R instead of S_{F_R}). Then $U \in S_F$ and S_F is a *semilattice*, i.e. $X, Y \in S_F$ implies $X \cap Y \in S_F$.

A set $X \in S_F$ is called (meet)-*irreducible* if $X = Y \cap Z$, $Y, Z \in S_F$ imply $X = Y$ or $X = Z$. The family of all irreducible sets is denoted by $GEN(F)$. Notice that the usual mathematical notation for $GEN(F)$ is $M(S_F)$, but we adopt the terminology of database theory here.

$GEN(F)$ is the unique minimal subfamily of generators in S_F such that each member of S_F can be expressed as an intersection of sets in $GEN(F)$ (where the set U is considered to be the intersection of an empty collection of sets).

It has been shown by Mannila and Räihä [MR86] that for a set F of FDs on U it holds that

$$GEN(F) = MAX(F) = \bigcup_{A \in U} MAX(F, A)$$

where $MAX(F, A) = \{Y \subseteq U \ : \ Y$ is a nonempty maximal set (with respect to $\subseteq$) such that $Y \to A \notin F^+\}$.

In [MR86] an algorithm is presented which computes an Armstrong relation R for a given FD-set F from $GEN(F)$ in time polynomial in the size of $GEN(F)$. On the other hand, if R is a given relation, then the MAX-sets for F_R, and hence also $GEN(F_R)$, can be computed in polynomial time (this follows easily from results in [BDFS84], [MR86], [MR87]).

Each $X \in MAX(F, A)$ can be written and interpreted as *excluded functional dependency (XFD)* with maximal left hand side, i.e., as an expression $X \not\to A$ such that $\forall B \in U-X : XB \rightarrow A$.

3 Some Results on Excluded Functional Dependencies

Excluded functional dependencies (in a similar way as MAX-sets) are just an alternative way of representing the information conveyed by a cover F of functional dependencies. When we speak about sets of excluded FDs we always assume that these FDs have single attributes as right hand sides, that the right hand side attribute of an XFD does not occur in the left hand side of the same XFD, and that all left hand sides corresponding to the same right hand side are maximal w.r.t. set inclusion, i.e., the set contains no pair of distinct XFDs $X \not\rightarrow A, Y \not\rightarrow A$, such that $X \subseteq Y$.

Excluded functional dependencies appear to be more intuitive than MAX-sets. However, when dealing with excluded FDs, some care has to be taken. If a set $\mathcal{X}$ of XFDs on a set of attributes U is given, we wish that this set represents *all* those dependencies which do not hold in a given situation. The corresponding set of all FDs which do hold is then represented by the cover:

$$F_{\mathcal{X}} = \{X \rightarrow A \; : \; X \subseteq U \wedge A \in U \wedge A \notin X \wedge \nexists Y \not\rightarrow A \in \mathcal{X} : X \subseteq Y\}.$$

Consider for example the set of excluded FDs $\mathcal{X} = \{AB \not\rightarrow C, AC \not\rightarrow B, B \not\rightarrow A, C \not\rightarrow A\}$ defined on a set of attributes $U = ABC$. Then $F_{\mathcal{X}} = \{BC \rightarrow A\}$. It is, however, important to note that there exist sets $\mathcal{X}$ of excluded FDs with maximal left hand sides, for which $F_{\mathcal{X}}$ is "unreasonable" because it implies FDs which should be forbidden (i.e. excluded) according to $\mathcal{X}$. The following example displays such a situation.

Consider a set $\mathcal{X}$ containing a single excluded FD $\mathcal{X} = \{B \not\rightarrow A\}$ defined on a set of attributes $U = ABC$. Then $F_{\mathcal{X}}$ is equivalent to the cover $\{C \rightarrow A, A \rightarrow B, C \rightarrow B, A \rightarrow C, B \rightarrow C\}$ Of course the FD $B \rightarrow A$ follows from $F_{\mathcal{X}}$; hence this FD is both excluded and requested. It can be seen that such situations arise when a set of excluded FDs is incomplete, in the sense that some necessary excluded FDs (in our case, for instance, $C \not\rightarrow A$ or $B \not\rightarrow C$) are missing. Let us therefore define the notion of *complete set of XFDs*.

A set $\mathcal{X}$ of excluded FDs is *complete* if $F_{\mathcal{X}}$ does not imply any excluded FD, i.e., if no FD $X \rightarrow A$ can be derived from $F_{\mathcal{X}}$, such that $X \not\rightarrow A \in \mathcal{X}$.

According to the semantics we give to sets of XFDs, only complete sets of XFDs make sense. Indeed, if a set of XFDs is incomplete, then it expresses that certain FDs are both forbidden and valid.

The following theorem relates complete XFD-sets to MAX-sets.

Theorem 1 *Let $\mathcal{X}$ be a set of XFDs defined on a set of attributes U. Let $RHS(\mathcal{X}, A) = \{X : X \not\rightarrow A \in \mathcal{X}\}$ for each $A \in U$. $\mathcal{X}$ is a complete set of XFDs iff $\forall A \in U : RHS(\mathcal{X}, A) = MAX(F_{\mathcal{X}}, A)$.*

Proof.

if. Assume that $\forall A \in U : RHS(\mathcal{X}, A) = MAX(F_{\mathcal{X}}, A)$. Each $MAX(F_{\mathcal{X}}, A)$, by definition, contains only sets of attributes which do not determine A w.r.t. $F_{\mathcal{X}}$. Thus there cannot be any FD $X \rightarrow A$ which follows from $F_{\mathcal{X}}$ such that X is equal to any element of $MAX(F_{\mathcal{X}}, A) = RHS(\mathcal{X}, A)$. Hence $\mathcal{X}$ is complete.

only if. Let $\mathcal{X}$ be a complete set of XFDs.

- We show that $\forall A \in U : RHS(\mathcal{X}, A) \subseteq MAX(F_{\mathcal{X}}, A)$.
  Assume that for some $A \in U$, $RHS(\mathcal{X}, A) \not\subseteq MAX(F_{\mathcal{X}}, A)$. Then there exists a XFD $X \not\rightarrow A \in \mathcal{X}$ such that $X \notin MAX(F_{\mathcal{X}}, A)$. X must be a (proper) subset of some element Y of $MAX(F_{\mathcal{X}}, A)$, otherwise $X \rightarrow A$ would hold, and $\mathcal{X}$ would not be complete. Thus there is an $Y \subseteq U$ with $XY \in MAX(F_{\mathcal{X}}, A)$ and $Y \neq \emptyset$ and $Y \cap XA = \emptyset$. On the other hand, since the XFD $X \not\rightarrow A$ of $\mathcal{X}$ has a maximal left hand side, it must hold by definition of $F_{\mathcal{X}}$ that $XY \rightarrow A \in F_{\mathcal{X}}$. This is in contradiction to $XY \in MAX(F_{\mathcal{X}}, A)$. We thus have shown that $RHS(\mathcal{X}, A) \subseteq MAX(F_{\mathcal{X}}, A)$.
- We show that $\forall A \in U : MAX(F_{\mathcal{X}}, A) \subseteq RHS(\mathcal{X}, A)$.
  Assume that for some $A \in U$, $MAX(F_{\mathcal{X}}, A) \not\subseteq RHS(\mathcal{X}, A)$. Then there exists $X \in MAX(F_{\mathcal{X}}, A)$ such that $X \notin RHS(\mathcal{X}, A)$. There are two cases to consider. In the first case X is not a subset of any element of $RHS(\mathcal{X}, A)$. Then $X \rightarrow A \in F_{\mathcal{X}}$. Contradiction to $X \in MAX(F_{\mathcal{X}}, A)$. In the second case, X is a proper subset of some $Y \in RHS(\mathcal{X}, A)$. Since $X \in MAX(F_{\mathcal{X}}, A)$ and Y is a proper superset of X the FD $Y \rightarrow A$ can be derived from $F_{\mathcal{X}}$; but $Y \not\rightarrow A$ is an excluded FD in $\mathcal{X}$. Thus $\mathcal{X}$ is not complete. Contradiction. Hence $MAX(F_{\mathcal{X}}, A) \subseteq RHS(\mathcal{X}, A)$.

The theorem is proved. □

If a set $\mathcal{X}$ of XFDs is complete, then an Armstrong relation R for $F_{\mathcal{X}}$ can be computed in polynomial time: Construct $GEN(F_{\mathcal{X}})$ by uniting all sets $RHS(\mathcal{X}, A)$ and then apply the polynomial algorithm of [MR86] to construct an Armstrong relation for $F_{\mathcal{X}}$ from $GEN(F_{\mathcal{X}})$. Note also that the cardinality of $F_{\mathcal{X}}$ can be exponential in the cardinality of $\mathcal{X}$.

Assume that a set $\mathcal{X}$ of XFDs on a set of attributes U is given. Assume furthermore that one has to compute the closure $C_{F_{\mathcal{X}}}(X)$ of a set of attributes $X \subseteq U$. One way is to compute first $F_{\mathcal{X}}$ and then use the CLOSURE algorithm as described in Section 2. However, this is not advisable since the size of $F_{\mathcal{X}}$ may be exponential in the one of $\mathcal{X}$. Fortunately there is a much simpler way of computing $C_{F_{\mathcal{X}}}(X)$. The following algorithm XFD-closure computes $C_{F_{\mathcal{X}}}(X)$ directly from $\mathcal{X}$ and X:

```
Algorithm XFD-CLOSURE
Input:  a set $\mathcal{X}$ of XFDs over $U$
        and a set $X \subseteq U$ of attributes.
Output: $C_{F_{\mathcal{X}}}(X)$
Method:
result := $X$;
WHILE there exists an attribute $A \in U$ such that
        $A \notin$ result AND
```

```
            there is no XFD $Y \not\rightarrow A \in \mathcal{X}$ such that $result \subseteq Y$
    DO $result := result \cup A$;
    RETURN($result$).
```

Theorem 2 *The* XFD-CLOSURE *algorithm applied to $\mathcal{X}, U$, and X effectively computes $C_{F_\mathcal{X}}(X)$.*

Proof. Let U be a set of attributes, let $A \in U$, and let $\mathcal{X}$ be a set of XFDs on U. By definition of $F_\mathcal{X}$, the following statements (1) and (2) are equivalent:

(1) there is a FD $Y \rightarrow A \in F_\mathcal{X}$

(2) there is no XFD $Z \not\rightarrow A$ in $\mathcal{X}$ such that $Y \subseteq Z$.

Now let *result* be an arbitrary subset of U. It follows that the following statements (1') and (2') are equivalent:

(1') there is a FD $Y \rightarrow A \in F_\mathcal{X}$ such that $Y \subseteq result$

(2') there is no XFD $Y \not\rightarrow A$ in $\mathcal{X}$ such that $result \subseteq Y$.

Indeed, (1') is equivalent to the statement $result \rightarrow A \in F_\mathcal{X}$ which in turn is equivalent to (2').

Now consider the XFD-CLOSURE algorithm for $\mathcal{X}$ and note that condition (2') occurs in the body of the algorithm. If we replace this condition with condition (1') we get exactly the body of the CLOSURE algorithm for $F_\mathcal{X}$. Hence the output of the XFD-CLOSURE algorithm is $C_{F_\mathcal{X}}(X)$. □

From the above theorem it follows that for each set $X \subseteq U$, $C_{F_\mathcal{X}}(X)$ can be computed in polynomial time from $\mathcal{X}$ and U. Moreover, the XFD-CLOSURE algorithm can be used as a tool for testing in polynomial time whether a given set $\mathcal{X}$ of XFDs is complete. Indeed, the following criterion follows trivially from the definition of completeness:

Completeness Criterion A *A set $\mathcal{X}$ of XFDs is complete iff for each XFD $X \not\rightarrow A \in \mathcal{X}$, $A \notin C_{F_\mathcal{X}}(X)$.*

Obviously, the test $A \notin C_{F_\mathcal{X}(X)}$ can be performed by using the XFD-CLOSURE algorithm.

Let us now derive a simple sufficient (but not necessary) condition for the completeness of a set $\mathcal{X}$ of XFDs:

Completeness Criterion B *A set $\mathcal{X}$ of XFDs is complete if for each XFD $X \not\rightarrow A \in \mathcal{X}$ and for each $B \in U - (XA)$ there is an XFD $Y \not\rightarrow B \in \mathcal{X}$ such that $X \subseteq Y$.*

Proof. Assume that Criterion B is satisfied. Let $X \not\rightarrow A$ be an XFD of $\mathcal{X}$. Note that the XFD-CLOSURE algorithm applied to $\mathcal{X}$ and X stops immediately with output X. Hence $C_{F_{\mathcal{X}}}(X) = X$. Therefore, by Completeness Criterion A, we conclude that $\mathcal{X}$ is complete. □

We will use this criterion in the proof of a theorem in Section 5.

Let us now make a remark which emphasizes the importance of the notion of completeness. Assume that an incomplete set of XFDs is given. We will show that such a set, in general, can be extended to several different (minimal) complete sets of XFDs. Hence incomplete sets of XFDs do not contain enough information for characterizing FD-families unambiguously. We will show this on hand of a simple example.

Consider again the set $\mathcal{X}$ containing a single excluded FD $\mathcal{X} = \{B \not\rightarrow A\}$ defined on a set of attributes $U = ABC$. We have already seen that this set is incomplete. We can extend $\mathcal{X}$ to a complete set either by enlarging the lhs of its XFD, yielding $\mathcal{X}_1 = \{BC \not\rightarrow A\}$, or by adding another XFD, yielding $\mathcal{X}_2 = \{B \not\rightarrow A, B \not\rightarrow C\}$. It can be easily seen by applying Completeness Criterion B that both $\mathcal{X}_1$ and $\mathcal{X}_2$ are complete. Of course $\mathcal{X}_1$ and $\mathcal{X}_2$ correspond to different sets of FDs $F_{\mathcal{X}_1}$ and $F_{\mathcal{X}_2}$. Furthermore, $\mathcal{X}_1$ and $\mathcal{X}_2$ are both minimally complete in the sense that any omission of an attribute or of an XFD would result in incompleteness.

We conclude this Section by making a few comments on related work. Excluded FDs are also studied by Thalheim in [Tha88] where their use for database design is motivated; moreover [Tha88] introduces the notion of *excluded multivalued dependency (XMVD)* and states derivation rules for FDs, MVDs, XFDs, and XMVDs. The notion of *functional independency* which is similar to the one of an XFD has been introduced by Janas [Ja88, Ja89]. Janas analyzes covers consisting of both, FDs and functional independencies. According to Janas, a set G of FDs and functional independencies is free of contradictions if there is no FD $X \rightarrow Y$ such that both $X \rightarrow Y$ and $X \not\rightarrow Y$ are implied by G. This concept seems to be close to the one of completeness; there is, however, a main difference between our approach and the one of Janas: We make the *closed world assumption* to sets of XFDs but Janas does not make this assumption for sets of functional independencies. For example, in the setting of Janas, the set $\{B \not\rightarrow A\}$ is free of contradictions, while in our setting this set is incomplete and thus expresses contradictory information.

4 Generating all Keys of a Relation Instance

The Dependency Inference Problem (Problem 2) is inherently exponential. Mannila and Räihä [MR87] show an example of a relation instance R containing $O(n)$ tuples, where $n = |U|$, such that there is a minimum cardinality cover F of F_R containing $O(2^{n/2})$ FDs. Nevertheless, a useful and practical algorithm for inferring dependencies from relation instances is developed in [MR87]. This algorithm has demonstrated a satisfactory efficiency when being used for "real-life" database design problems.

We will now show that the problem of finding all keys of a relation instance can be polynomially transformed to the Dependency Inference Problem. This transformation is useful because it allows to use highly practical algorithms for dependency inference (such as the one presented in [MR87]) for generating all keys to a given relation instance.

As a by-product of our polynomial transformation we also get a new proof for the exponential complexity of dependency inference. This complexity result follows directly from our transformation and from a well known result on the complexity of key-generation. Consider the following algorithm.

Algorithm *Input:* a relation $R = \{t_1, \ldots, t_m\}$ over U.
Output: a set F of FDs.

Step 1. Find the *equality set* $E_R = \{E_{ij} : 1 \leq i < j \leq m\}$, where $E_{ij} = \{A \in U : t_i(A) = t_j(A)\}$.

Step 2. Find the maximal sets among $E_R - \{U\}$. Denote them by $X_1, \ldots, X_p$.

Step 3. Construct a family $\{X_i - A : A \in U, i = 1, \ldots, p\}$ and denote its elements by $Y_1, \ldots, Y_r$. Suppose $Y_0 = U$.

Step 4. Construct a relation $R' = \{t'_0, ..., t'_r\}$ where

$$t'_i(A) = \begin{cases} 0 & \text{if } A \in Y_i \\ i & \text{otherwise, } A \in U, i = 1, ..., r \end{cases}$$

Step 5. Using the algorithm for solving the dependency inference problem, find a cover F' of $F_{R'}$.

Step 6. Find a minimum cover F of F'.

Clearly, all the steps except step 5 require polynomial time in $|R|$, that is, in $n \cdot m$. For a discussion and characterization of the equality sets E_R and E_{ij} see [DT88].

Theorem 3 *The output F of the above algorithm consists of FDs $K_1 \to U, \ldots, K_l \to U$, where $K_1, \ldots, K_l$ are all the minimal keys of R.*

Proof. According to [DT88], $X_1, \ldots, X_p$ are so-called *antikeys*, i.e. maximal non-keys. According to [MR86], R' is a relation whose antikeys are $X_1, \ldots, X_p$ and by [BDK, theorem 3] the families of keys of R and R' coincide. Moreover, by [DHLM89] $F_{R'}$ is in BCNF, and hence its minimum cover consists of FDs $K_i \to U$ for $K_i, i = 1, \ldots, l$, the minimal keys of R'. □

It is shown in [MR87] that in many cases the algorithm solving dependency inference problem may work efficiently. In these case one can use the above algorithm to find the minimal keys of a relation. Remind, that this problem is inherently exponential as the number of keys of a given relation instance can be exponential in the size of the instance [BDFS84,DT87]. The last mentioned fact together with theorem 3 implies

Corollary 1 *The dependency inference problem has exponential complexity.* □

5 Deciding $F_R \subseteq F^+$ is Co-NP-Complete

In this Section we turn our attention to Problem 3. It is possible to show that this problem (FD-Relation Implication Problem) is co-$\mathcal{NP}$-complete. In order to do this, we will first define another problem and prove its co-$\mathcal{NP}$-completeness and then show the polynomial transformability of that problem to our problem.

The problem we will first consider can be described as follows:

> **Name:** SUBSET DELIMITER COMPLEMENTARITY (SDC)
> **Instance:** a finite set S, a collection $G_1 \ldots G_n$ of subsets of S, and a collection $D_1 \ldots D_m$ of subsets of S.
> **Question:** Is it true that $\forall X \subseteq S$: ($(\exists i, 1 \leq i \leq n : G_i \subseteq X)$ or $(\exists j, 1 \leq j \leq m : X \subseteq D_j)$) ?

In order to show the co-$\mathcal{NP}$-completeness of SDC, we will use the MONOTONE 3SAT problem which is known to be $\mathcal{NP}$-complete [Go78, GJ79]:

> **Name:** MONOTONE 3SAT (M3SAT)
> **Instance:** a finite set U of propositional variables and a collection C of clauses over U such that each clause contains exactly three literals and each clause contains either only negated or only un-negated literals.
> **Question:** Is there a satisfying truth assignment for C ?

Theorem 4 *The SDC Problem is co-$\mathcal{NP}$-complete.*

Proof. It is easy to see that the problem is in co-$\mathcal{NP}$. In order to show that its solution is negative, guess a subset $Z \subseteq S$ nondeterministically such that Z is neither a superset of any G_i nor a subset of any D_j.

Let us now show that the complement of M3SAT can be reduced polynomially to our problem. Consider an instance (U, C) of M3SAT. Assume without loss of generality that C consists of k clauses $C_1 \ldots C_k$ such that the first n clauses are positive and the remaining m clauses are negative (with $m = k - n$).

We construct an instance of the SDC problem from (U, C) as follows. Let $S = U$. For each $1 \leq j \leq n$ let $D_j = U - C_j$ and for each $1 \leq i \leq m$ let $G_i = \{p : \neg p \in C_{n+i}\}$. Clearly the D_j and G_i can be constructed in polynomial time from C.

In the sequel of this proof, any truth value assignment for the propositional variables of U is represented as the subset of U consisting of all those propositional variables which are assigned "true".

C is unsatisfiable, iff for each truth value assignment $\tau \subseteq U$ there exists a clause C_i, $1 \leq i \leq k$ such that C_i is falsified by τ. In particular:

• A positive clause $C_j \in C$ is falsified by τ iff no propositional variable appearing in τ also appears in C_j, i.e., iff $\tau \subseteq U - C_j = D_j$.

• A negative clause $C_i \in C$ is falsified by τ iff all propositional variables occurring in C_i (in negated form) have truth value "true" under τ, i.e., iff $G_{i-n} \subseteq \tau$.

Thus C is unsatisfiable iff for each $\tau \subseteq S$, it holds that $(\exists i, 1 \leq i \leq n : G_i \subseteq \tau)$ or $(\exists j, 1 \leq j \leq m : \tau \subseteq D_j)$. We thus have polynomially transformed the complement of the M3SAT problem to the SDC problem. This completes our proof. □

The following Corollary shows the co-$\mathcal{NP}$-completeness of a slightly stronger version of the SDC problem.

Corollary 2 *The SDC problem remains co-$\mathcal{NP}$-complete even if it is restricted to those instances for which the family of sets D_j is an antichain, i.e., no D_j is a subset of a D_i, for $i \neq j$ and $1 \leq i, j \leq m$.*

Proof. Consider an instance of SDC whose sets D_j do not form an antichain. By eliminating all those D_j which are contained in any other D_i, we get an equivalent instance satisfying our restriction. Of course this transformation can be done in polynomial time. □

We are now ready for proving our complexity result for Problem 2.

Theorem 5 *It is co-$\mathcal{NP}$-complete to decide whether for a given relation (instance) R and for a given set F of FDs it holds that $F_R \subseteq F^+$.*

Proof. Clearly the problem is in co-$\mathcal{NP}$. Indeed, in order to show that $F_R \not\subseteq F^+$ it is sufficient to guess nondeterministically an FD which is in F_R (testable in polynomial time) but which is not in F^+ (again testable in polynomial time). Let us now show completeness in co-$\mathcal{NP}$.

Consider an instance of the SDC problem consisting of a set S and of families of subsets $G_1 \dots G_n$ and $D_1 \dots D_m$. According to Corollary 2 we may assume that the sets $D_1 \dots D_m$ form an antichain.

From this instance we will construct a set F of FDs and a set $\mathcal{X}$ of XFDs as follows. Let us view the elements of S as attributes and consider a new attribute $A \notin S$. In the sequel of this proof, all FDs and XFDs are defined on the set of attributes $S' = S \cup \{A\}$.

Let $F \;=\; \{G_i \rightarrow A \;:\; 1 \le i \le n\}$ and
let $\mathcal{X} \;=\; \{D_j \not\rightarrow A \;:\; 1 \le j \le m\} \cup \{(S' - B) \not\rightarrow B \;:\; B \in S\}$.

Note that the set $F_{\mathcal{X}}$ contains only FDs with right hand side A. More precisely, $F_{\mathcal{X}}$ consists of all FDs of the form $X \rightarrow A$ such that $X \subseteq S$ and $X \not\subseteq D_j$ for $1 \le j \le m$. Furthermore, $F_{\mathcal{X}}^+$, besides the trivial FDs over S', contains exactly the FDs of $F_{\mathcal{X}}$. (This follows from the fact that the pseudotransitivity rule cannot be applied to the FDs of $F_{\mathcal{X}}$ in order to generate new nontrivial FDs.)

On the other hand, the set F^+ consists of all FDs $X \rightarrow A$ such that X is a superset of some G_i with $1 \le i \le n$ plus the trivial FDs over S'.

From these observations it follows that $F_{\mathcal{X}}^+ \subseteq F^+$ iff each subset of S which is not a subset of any D_j is a superset of some G_i. In other words, $F_{\mathcal{X}}^+ \subseteq F^+$ iff our SDC Problem-instance has a positive solution.

Since the D_j $(1 \le j \le m)$ form an antichain, the XFDs of $\mathcal{X}$ all have maximal left hand sides. Moreover, the set $\mathcal{X}$ of XFDs satisfies the Completeness Criterion B of Section 3. Hence $\mathcal{X}$ is complete and a relation instance R can be found in polynomial time such that $F_R = F_{\mathcal{X}}^+$. Now our SDC problem instance has a positive solution iff $F_R \subseteq F^+$.

We thus have shown how an instance of the SDC problem can be transformed into an instance of the FD-Relation Implication Problem (Problem 3). It is immediately verifiable that this transformation can be performed in polynomial time in the size of the given SDC instance. It follows that Problem 3 is co-$\mathcal{NP}$-complete. □

Of course, the converse problem, that is, to check up if $F^+ \subseteq F_R$, can be solved in polynomial time. However, as pointed out in the introduction, it is still unknown if the problem 4 (FD-Relation Equivalence Problem) is polynomially solvable or not. Here we show that if F does not contain FDs with small left-hand sides then both problems 3 and 4 can be solved in polynomial time.

Proposition 1 *Suppose for each $X \rightarrow Y \in F$ one has $|U| - |X| \le k$, where k is a* *constant. Then both problems 3 and 4 can be solved in polynomial time.*

Proof. Given a relation instance R and a set $X \subseteq U$, to find $C_R(X)$ requires polynomial time in $|R|$. Hence we can check in polynomial time if $C_R(X) = X$ for all X with $|U| - |X| = k - 1$. Since S_R is a semilattice, for each nontrivial FD $X \to Y \in F_R$ it holds that $|U| - |X| \leq k$. Therefore, to make sure that $F_R \subseteq F^+$, we just have to consider all sets X with $|U| - |X| \leq k$ (there are less than $|U|^k$) and to check that $C_R(X) \subseteq C_F(X)$. □

6 Complexity of the Main Problems : Special Cases

As it has been shown at the end of the previous section, the problem which is generally co-$\mathcal{NP}$-complete can be solved in polynomial time if some additional properties hold. This fact leads us to the idea to study several special types of relation schemes in order to find out if problems 1-4 are polynomial for these relation schemes.

In this section we are going to study three types of relation schemes. All these types have already been investigated more or less widely. We formulate the properties for a relation scheme $< U, F >$ and for its associated closure L_F and semilattice S_F.

Property 1 *There is a cover of F consisting of unary FDs, i.e. of FDs of type $A \to B, A, B \in U$.*

Property 2 *There is a cover of F of type $\{X_1 \to A_1, \ldots, X_r \to A_r\}$ such that $X_1 \subseteq \ldots \subseteq X_r$.*

Property 3 *A relation scheme $< U, F >$ is in BCNF.*

The properties 1 and 3 seem to be simply explained from the practical point of view, note that property 3 is very desirable. Property 2 is interesting from a mathematical point of view because it corresponds to a relevant class of semilattices and closures.

First, we establish the equivalent formulations of the main properties.

Proposition 2 *Given a relation scheme $< U, F >$, the following are equivalent:*
1) $< U, F >$ satisfies property 1,
2) C_F is topological, i.e. $C_F(X \cup Y) = C_F(X) \cup C_F(Y)$,
3) S_F is a distributive lattice.

The proof is straightforward. □

Proposition 3 *([DLM89]) Given a relation scheme $< U, F >$, the following are equivalent:*

1) $< U, F >$ satisfies property 2,
2) C_F is separatory, that is, if $C_F(X) \neq X$ and $C_F(Y) \neq Y$, then $C_F(X \cap Y) \neq X \cap Y$,
3) S_F is separatory, that is, $\mathbf{2}^U - S_F$ is a semilattice again. □

Proposition 4 *([DHLM89]) Given a relation scheme $< U, F >$, the following are equivalent:*
1) $< U, F >$ satisfies property 3,
2) For each $X \subseteq U$ either $C_F(X) = X$ or $C_F(X) = U$,
3) $S_F - \{U\}$ is an ideal of $\mathbf{2}^U$, i.e. if $Y \subseteq X \in S_F - \{U\}$, then $Y \in S_F$. □

Further we will show that some considered problems can be solved in polynomial time if it is known that a relation scheme satisfies property 1 or 2 or 3. Hovewer, in order to use an algorithm solving a problem in a special case one has to make sure that either scheme or relation satisfies the required property. Therefore, it would be desirable if all the properties 1-3 could be recognized in polynomial time. The next Theorem shows that this fact holds.

Theorem 6 *All the properties 1-3 are polynomially recognizable for both relation schemes and relations.*

Proof. Property 1. a) for relation schemes. It is almost obvious that unary FDs cannot be derived from other FDs. Hence, a relation scheme satisfies property 1 iff a nonredundant cover of F consists of unary FDs only.

b) For relations. Given a relation R, we can find $GEN(F_R)$ in polynomial time in $|R|$, see [DT88]. Let us first prove that F_R satisfies property 1 iff $X \cup Y \in S_R$ for every $X, Y \in GEN(F_R)$. Really, if F_R satisfies property 1, then it follows from proposition 2 that $X \cup Y = C_R(X) \cup C_R(Y) = C_R(X \cup Y)$ and $X \cup Y \in S_R$. Conversely, if $X \cup Y \in S_R$ for every $X, Y \in GEN(F_R)$, consider arbitrary $V, W \in S_R$. Suppose $V = X_1 \cap \ldots \cap X_k, W = Y_1 \cap \ldots \cap Y_l$, where $X_1, \ldots, X_k, Y_1, \ldots, Y_l \in GEN(F_R)$. Then $V \cup W = (X_1 \cap \ldots \cap X_k) \cup (Y_1 \cap \ldots \cap Y_l) = \bigcap_{i=1}^{k} \bigcap_{j=1}^{l} (X_i \cup Y_j) \in S_R$, i.e. C_R is topological. Since to find a closure C_R requires polynomial time, the above property can be checked polynomially.

Property 2. a) For relation schemes. First we prove that if a relation scheme $< U, F >$ satisfies property 2 and $X \to A, Y \to B \in F^+$ then either $X \cap Y \to A \in F^+$ or $X \cap Y \to B \in F^+$, where $A \notin X, B \notin Y$. Really, if it is not true, then $A, B \notin C_F(X \cap Y)$. Hence, both $X \cup C_F(X \cap Y)$ and $Y \cup C_F(X \cap Y)$ are nonclosed, and by proposition 3 $C_F(X \cap Y) = (X \cup C_F(X \cap Y)) \cap (Y \cup C_F(X \cap Y))$ is nonclosed, a contradiction.

Suppose without loss of generality that F consists of FDs $X \to A$, where A is an attribute. Hence, if a relation scheme satisfies property 2, for every two FDs $X \to A, Y \to B \in F$ either $(F - \{X \to A\}) \cup \{X \cap Y \to A\}$ or $(F - \{Y \to B\}) \cup \{X \cap Y \to B\}$ is a cover of F. Since the membership problem for FDs is polynomial [Ma83], we need only the following to finish the proof: if we are given a family $\mathcal{A} = \{X_1, \ldots, X_k\}$ of subsets of U, and by one step we can change either X_i

or X_j to $X_i \cap X_j$, then $\mathcal{A}$ can be transformed to a chain by a polynomial number of steps.

First we show how to transform $\mathcal{A}$ to $\mathcal{A}' = \{X'_1, \ldots, X'_k\}$ where $X'_i = X_i$ for some i and $X'_j \subseteq X'_i$ for all $j \neq i$. We use induction on k.

If $\mathcal{A}$ contains unique maximal element X_i, we are done. If X_i, X_j are two maximal elements of $\mathcal{A}$, consider $\mathcal{A} - \{X_i\}$ and transform it to $\mathcal{A}^0 = \{X_l^0 : l \neq i\}$ where $X_p^0 = X_p$ for some p and $X_l^0 \subseteq X_p^0$ for all $l \neq i$. If $X_p \subseteq X_i$, we are done. If X_i and X_p are incomparable, consider all the pairs $\{X_i, X_l^0\}, l \neq i$. If for some l we can change X_i to $X_i \cap X_l^0$, then $\mathcal{A}' = \mathcal{A}^0 \cup \{X_i \cap X_l^0\}$. If for all the pairs we can only change X_l^0 to $X_i \cap X_l^0$, then $\mathcal{A}' = \{X_i\} \cup \{X_i \cap X_l^0 : l \neq i\}$.

If $k = 2$, it takes one step to transform $\mathcal{A}$ to a chain. Since each ith iteration takes no more than i additional steps, it takes $O(k^2)$ steps to transform $\mathcal{A}$ to $\mathcal{A}'$. Then, if we apply the above algorithm to $\mathcal{A}' - \{X_i\}$ etc, we obtain a chain by no more than $k-1$ iterations. Hence, $\mathcal{A}$ can be transformed to a chain by $O(k^3)$ steps being used. This shows the polynomiality of the recognition of property 2 for relation schemes.

b) For relations. It follows immediately from proposition 3 that if F_R satisfies property 2, then all the elements of $GEN(F_R)$ have cardinality $n, n-1$ or $n-2$. Moreover, S_R is separatory if and only if the matrix $a = \| a_{ij} \|, i, j = 1, \ldots, n$:

$$a_{ij} = \begin{cases} 1 & \text{if } U - \{A_i, A_j\} \in S_R \\ 0 & \text{otherwise,} \end{cases}$$

where $U = \{A_1, \ldots, A_n\}$ is ***absolutely determined***, that is, each submatrix of a has a saddle point [GL90]. The last property can be checked in time $O(n^4)$ [GL90].

Property 3. a) For relation schemes. It is wellknown that the BCNF property of relation schemes can be tested in polynomial time. It can be shown, for instance, as follows. It is almost evident that a relation scheme $< U, F >$ is in BCNF iff its minimum cover consists of FDs $\{K_i \to U, i = 1, \ldots, l\}$, where $K_i, i = 1, \ldots, l$, are the minimal keys of $< U, F >$. Since to find a minimum cover takes polynomial time [Ma83], and testing whether a set of arttributes is a minimal key also takes polynomial time, BCNF can be recognized in polynomial time.

b) For relations. See [DHLM89] for a polynomial algorithm.

The proof is complete. □

Now we are ready to present the main result about the complexity of problems 1-4 if it is known that a relation scheme $< U, F >$ (or $< U, F_R >$ if input is R) satisfies additional properties.

Theorem 7 *The problems 1-4 can be solved in polynomial time if it is known that a relation scheme $< U, F >$ (for problems 1,3,4) or $< U, F_R >$ (for problem 2) satisfies property 1 or 2.*

Proof. Property 1. The polynomiality of constructing Armstrong relation was proved in [MR89], the polynomiality of the other problems is almost evident.

Property 2. a) Problem 1. According to the proof of previous theorem (see also [GL90]) $GEN(F)$ can be computed in polynomial time. Applying algorithm of [MR86, p.136], we find an Armstrong relation.

b) Problem 2. We use the concepts of $nec(A)$ and $gendep(A)$ (see [MR87]). Let $R = \{t_1, \ldots, t_n\}$ be a relation over U. Let $disag(i,j) = \{A \in U : t_i(A) \neq t_j(A)\}$ and $nec(A) = \{disag(i,j) - A : A \in disag(i,j)\}$. Suppose $gendep(A) = \{\{A_1, \ldots, A_r\} \rightarrow A : A_i \in X_i, i = 1, \ldots, r\}$, where $nec(A) = \{X_1, \ldots, X_r\}$. Then $\bigcup\{gendep(A) : A \in U\}$ is a cover of F_R. Suppose $X_A = \bigcap(\{A_1, \ldots, A_r\} : A_i \in X_i, i = 1, \ldots, r)$. If a relation scheme $< U, F_R >$ satisfies property 2, it follows from the proof of Theorem 6 that $\{X_A \rightarrow A : A \in U\}$ is a cover of F_R. Clearly, $B \in X_A$ iff $\{B\} = X_i$ for some $X_i \in nec(A)$, and $nec(A)$ can be computed in polynomial time. Therefore, it takes polynomial time to find a cover of F_R.

c) Problems 3-4. According to [DLM89], $F_R \subseteq F^+$ iff $S_F \subseteq S_R$, or iff $GEN(F) \subseteq S_R$. Since $GEN(F)$ can be computed in polynomial time, the checking of the last condition takes polynomial time too.

The theorem is completely proved. □

Property 1 can be easily generalized if we allow FDs $X \rightarrow A$ with $|X| < k$, $k > 1$. However, as the following theorem shows, it is impossible to get a polynomiality result for Problem 1 w.r.t. such relation schemes.

Proposition 5 *Problem 1 has exponential complexity even if it is known that a relation schema $< U, F >$ satisfies the property: for each FD $X \rightarrow A \in F^+$ there is an FD $Y \rightarrow A \in F^+$ with $Y \subseteq X$ and $|Y| < k$, $k > 1$.*

Proof. In [BDFS84] an example of a RS with $k = 2$ was constructed that satisfies the above property and provides a minimal Armstrong relation exponential in the number of FDs. □

Finishing this section, we discuss the complexity of the main problems for relations and relation schemes in BCNF.

Let $< U, F >$ be a relation scheme in BCNF. We can think without loss of generality that F consists of FDs $K_i \rightarrow U, i = 1, \ldots, l$, where $K_i, i = 1, \ldots, l$, are the minimal keys (if not, we compute a minimum cover in polynomial time). Let R be an Armstrong relation for $< U, F >$. Then we can find *antikeys*, that is, maximal nonkeys [Thi86], in polynomial time in $|R|$, see [DT88]. Conversely, if we have the family of antikeys, we can construct an Armstrong relation for $< U, F >$ according to the algorithm of section 2. Thus, we obtain

Proposition 6 *Problem 1 for relation schemes in BCNF is polynomially equivalent to finding the antikeys of a family of minimal keys.* □

The last problem was discussed in [Thi86]. The problem is inherently exponential.

Hovewer, it can be solved in polynomial time, with some additional conditions being added.

Proposition 7 *Problem 1 for relation schemes in BCNF can be solved in polynomial time if the number of minimal keys is bounded by a constant.*

Proof. It follows from [Thi86] and proposition 6. □

Now we prove an auxiliary result.

Proposition 8 *Problem 2 can be solved in polynomial time if the number of tuples of a relation is bounded by a constant.*

Proof. Let m be the number of tuples of a relation R. Then $nec(A)$ contains no more that m^2 sets (see the proof of theorem 7), and $gendep(A)$ has no more that n^{m^2} FDs. Hence, a cover of F_R can be computed in polynomial time. □

Corollary 3 *If the number of tuples of a relation is bounded by a constant, it takes polynomial time to find all its minimal keys.*

Proof. According to [DT88], the number of antikeys is no more than m^2, where m is the number of tuples of R. Hence, the number of minimal keys is no more than $n \cdot m^2$. By proposition 8, we can compute a cover of F_R in polynomial time, and, by [LO78], given a relation scheme, we can find its minimal keys in polynomial time in size of input and output. Hence, the minimal keys of R can be found in polynomial time. □

Now we immediately obtain from theorem 6, proposition 7, and corollary 3:

Proposition 9 *The problems 3 and 4 can be solved in polynomial time for a relation scheme in BCNF if either the number of minimal keys or the number of tuples of a relation is bounded by a constant.* □

We can demonstrate another example providing the problem 4 to be polynomial for relation schemes in BCNF. Remind that an antichain $\mathcal{A}$ is called *saturated* [BDK87,Thi86] if $\mathcal{A} \cup \{X\}$ is not antichain for every $X \notin \mathcal{A}$.

Proposition 10 *Let $< U, F >$ be a relation scheme in BCNF and R a relation in BCNF. If either the family of minimal keys of $< U, F >$ or the family of antikeys of R is suturated, the problem 4 can be solved in polynomial time.*

Proof. Let the family $\{K_1, \ldots, K_l\}$ of the minimal keys of $< U, F >$ be saturated. Find in polynomial time the family $\{X_1, \ldots, X_r\}$ of antikeys of R [DT88]. Then

$\{X_1,\ldots,X_r\}$ is the family of antikeys of $\{K_1,\ldots,K_l\}$ iff for all $i = 1,\ldots,l$, K_i is a minimal set that is not contained in some $X_j, j = 1,\ldots,r$. Clearly, the last condition can be checked in polynomial time. If the family of antikeys of R is saturated, the proof is the same. □

Several criteria providing the families of minimal keys and antikeys to be saturated are established in [Thi86].

7 Conclusion

In this paper we have investigated several aspects of Armstrong relations, dependency inference, and excluded functional dependencies. In particular, we have characterized those sets of excluded dependencies which effectively correspond to sets of FDs (and hence to Armstrong relations). We have shown that the problem of findings all minimal keys of a given relation instance can be solved by using practical algorithms for dependency inference. We proved that the problem whether all FDs that are valid in a given relation instance R do follow from a given cover F is co-$\mathcal{NP}$-complete. Finally, we have analyzed several conditions under which the main problems become polynomially solvable.

One relevant problem remains open: given a relation instance R and a cover F of FDs, what is the complexity of deciding whether $F_R = F^+$? This problem is important; it can be reformulated as follows: what is the complexity of recognizing that a given relation is an Armstrong relation for a given set of FDs. We plan to dedicate further research to this problem.

ACKNOWLEDGMENTS. The authors are grateful to Maddalena Boschetti, Thomas Eiter, and Ernesto Noce for useful comments and corrections to the first version of the manuscript.

References

[BDFS84] C.Beeri, M.Dowd, R.Fagin and R.Statman,
On the structure of Armstrong relations for functional dependencies,
J.Assoc. Comput. Mach. **31** (1984), 30-46.

[BDK87] G.Burosch, J.Demetrovics and G.O.H.Katona,
The poset of closures as a model of changing databases,
Order **4** (1987), 127-142.

[DHLM89] J.Demetrovics, G.Hencsey, L.O.Libkin and I.B.Muchnik,
Normal form relation schemes : a new characterization, *Manuscript*.

[DLM89] J.Demetrovics, L.O.Libkin and I.B.Muchnik,
Functional dependencies and the semilattice of closed classes,
MFDBS 89, Springer LNCS **364** (1989), 136-147.

[DT87] J.Demetrovics and V.D.Thi, Keys, antikeys and prime attributes, *Annales Univ. Sci. Budapest Sect. Comp.* **8** (1987), 35-52.

[DT88] J.Demetrovics and V.D.Thi, Some results about functional dependencies, *Acta Cybernetica* **8** (1988), 273-278.

[FA82] R. Fagin, Horn Clauses and Database Dependencies, *Journal of the ACM* **29:4** (1982), 952-985.

[GJ79] M.R. Garey and D.S. Johnson, Computers and Intractability - A Guide to the Theory of NP-Completeness, Freeman and Company, New York, 1979.

[Go78] E.M. Gold, Complexity of Automaton Identification from Given Data, *Information and Control*, **37** (1978), 302-320.

[GL90] V.A.Gurvich and L.O.Libkin, Absolutely determined matrices, to appear in *Math. Soc. Sci.*

[Ja88] J.M.Janas, On Functional Independencies, In: *Foundations of Software Technology and Theoretical Computer Science*, K.V. Nori and S. Kumar Eds., Springer LNCS *338* (1988) 487-508.

[Ja89] J.M.Janas, Covers for Functional Independencies, In: *Proceedings of the MFDBS 89 Conference*, J.Demetrovics and B. Thalheim Eds., Springer LNCS **364** (1989) 254-268.

[LO78] C.L.Lucchesi and S.L.Osborn, Candidate keys for relations, *J. of Computer and System Sciences* **17** (1978), 270-279.

[Ma83] D.Maier, *"The Theory of Relational Databases"*, Comp.Sci.Press, Rockville, MD, 1983.

[MR86] H.Mannila and K.-J.Räihä, Design by example: an application of Armstrong relations, *J. of Computer and System Sciences* **33** (1986), 126-141.

[MR87] H.Mannila and K.-J.Räihä, "Algorithms for Inferring Functional Dependencies" (Extended Abstract), Proceedings of the Thirteenth International Conference on Very Large Data Bases, Brighton, September 1987; Full paper submitted for publication.

[MR89] H.Mannila and K.-J.Räihä, Practical algorithms for findiding prime attributes and testing normal forms, *PODS 89*, pp. 128-133.

[MR90] H.Mannila and K.-J.Räihä, On the Complexity of Inferring Functional Dependencies, manuscript, submitted for publication, 1990.

[PBGV89] J.Paredaens, P.De Bra, M.Gyssens and D.Van Gucht, The Structure of the Relational Database Model, Springer-Verlag, Berlin, 1989.

[Tha88] B. Thalheim, Logical Relational Database Design Tools Using Different Classes of Dependencies, *J. of New Generation Comput. Syst*, *1*:3 (1988), 211-228.

[Thi86] V.D.Thi, Minimal keys and antikeys, *Acta Cybernetica* **7** (1986), 361-371.

(Received May 10, 1990)

Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1990

The complexity of a counting finite state automaton

C. A. Rich[1] and G. Slutzki[2]

Abstract

A counting finite-state automaton is a nondeterministic finite-state automaton which, on an input over its input alphabet, (magically) writes in binary the number of accepting computations on the input. We examine the complexity of computing the counting function of an NFA, and the complexity of recognizing its range as a set of binary strings. We also consider the pumping behavior of counting finite-state automata. The class of functions computed by counting NFA's

(1) includes a class of functions computed by deterministic finite-state transducers;
(2) is contained in the class of functions computed by polynomially time- and linearly space-bounded Turing transducers;
(3) includes a function whose range is the composite numbers.

1. Introduction

A counting finite-state automaton is a nondeterministic finite-state automaton which, on an input over its input alphabet, (magically) writes in binary the number of accepting computations on the input. The counting finite-state automaton—or counting NFA—is a finite-state analogue of the counting Turing Machine of Valiant [7].

It is known that the class #P of functions computed by polynomially time-bounded counting TMs includes the class FP of functions computed by polynomially time-bounded Turing transducers; however, it is not known if this inclusion is proper. Valiant [7,8] has shown several functions to be complete for #P, and these functions in #P are not computable in polynomial time if $P \neq NP$. These results suggest that FP is properly included in #P.

[1] Computer Science Department, California State Polytechnic University, Pomona, Pomona, CA 91768-4034, USA

[2] Computer Science Department, Iowa State University, Ames, IA 50011, USA

We consider finite-state analogues of these questions. We show that the class #NFA of functions computed by counting NFAs includes a class #DFT of counting functions computed by deterministic finite-state transducers. Although it is not known whether FP $\neq$ #P, we show that #DFT is properly included in #NFA by exhibiting a counting NFA whose range as a set of binary strings is not context-free, whereas the ranges of deterministic finite-state transducers are regular [2]. While some functions in #P are apparently not computable in polynomial time, we show that functions in #NFA can be computed using time polynomial and space linear in the the length of the input.

Since functions in #DFT have ranges which are regular and functions in #NFA have ranges which are not necessarily context-free, it is natural to investigate the complexity of counting NFA ranges. Intuitively, one might expect the range of a counting NFA to be efficiently recognizable simply because it is a finite-state model, but that is apparently not the case. We establish an upper bound by showing that the range of a counting NFA is recognizable nondeterministically using space linear in the length of the input, i.e., a context-sensitive language. We suggest an intractable lower bound by showing that the composite numbers—which are not known to be in P—are the range of a counting NFA.

In §2, we give notational conventions and formally define the counting function of an NFA. In §3, we show that the counting functions computed by deterministic finite-state transducers are properly included among those computed by nondeterministic finite-state automata, and give a counting NFA whose range is not context-free. In §4, we examine the complexity of computing the counting function of an NFA, and the complexity of recognizing its range as a set of binary strings. In §5, we consider the pumping behavior of a counting finite-state automaton. For a fixed input string, we show that the number of accepting computations—considered as a function of the number of times a fixed substring is pumped—satisfies a homogeneous linear recurrence equation of finite degree having integer coefficients.

2. Preliminary Definitions

In this section, we present notational conventions and our notions of counting function computed by finite-state automata. A *string* x is a finite sequence of *symbols* from a finite *alphabet*. The *length* of x, denoted $|x|$, is the number of symbols composing x. The *empty string*, denoted ϵ, is the string having length 0. The *concatenation* of two strings x and y is the string consisting of the symbols of x followed by the symbols of y, denoted xy. A *language* L is a set of strings over an alphabet, and $\|L\|$ denotes the cardinality of L. The empty set is denoted by ϕ; the set of integers $\{\ldots,-1,0,1,\ldots\}$ is denoted by $\mathcal{Z}$; and the set of natural numbers $\{0,1,2,\ldots\}$ is denoted by $\mathcal{N}$.

In this work, we frequently consider natural numbers as binary strings and vice versa. Formally, these conversions are functions $s\colon \mathcal{N} \to \{0,1\}^*$ and $\#\colon \{0,1\}^* \to \mathcal{N}$ defined by

$$s(k) = \text{the binary representation of } k \text{ without leading zeroes,}$$
$$\#(x) = \text{the number represented in binary by } x.$$

Note that $s(0) = \epsilon$. We extend s and # to sets of natural numbers and binary strings in the usual way by defining $s(K) = \{\, s(k) \mid k \in K \,\}$, and $\#(L) = \{\, \#(x) \mid x \in L \,\}$.

A *nondeterministic finite automaton (NFA)* is a 5-tuple $M = (Q, \Sigma, \delta, I, F)$, where Q is a finite set of *states*; Σ is a finite *input alphabet*; δ is a *transition function* from $Q \times \Sigma^*$ to subsets of Q; $I \subseteq Q$ is a set of *initial states*; and $F \subseteq Q$ is a set of *final states*. The *counting function* $\#\delta: Q \times \Sigma^* \to \mathcal{N}$ is defined recursively by

$$\#\delta(q,\epsilon) = \begin{cases} 1, & \text{if } q \in F; \\ 0, & \text{if } q \notin F, \end{cases}$$
$$\#\delta(q,\sigma x) = \sum_{p \in \delta(q,\sigma)} \#\delta(p,x).$$

The *counting function* $\#M: \Sigma^* \to \mathcal{N}$ is defined by

$$\#M(x) = \sum_{q \in I} \#\delta(q,x).$$

Intuitively, the counting function $\#M(x)$ ($\#\delta(q,x)$) is the number of accepting computations of M on input x (starting from state q). We extend the counting functions to languages $L \subseteq \Sigma^*$ in the usual way by defining $\#\delta(q,L) = \{\, \#\delta(q,x) \mid x \in L \,\}$, and $\#M(L) = \{\, \#M(x) \mid x \in L \,\}$. We consider the range of a counting NFA M to be the set of binary strings $s(\#M(\Sigma^*))$. The class of counting functions of NFAs is defined by $\#\text{NFA} = \{\, \#M \mid M \text{ is an NFA} \,\}$, and the class of their ranges is defined by $\text{range}(\#\text{NFA}) = \{\, s(\#M(\Sigma^*)) \mid M \text{ is an NFA with input alphabet } \Sigma \,\}$.

We also want to consider a deterministic counterpart of the counting NFA which produces a binary string by transduction rather than counting accepting computations. Intuitively, our deterministic finite-state transducer is a special case of the deterministic Generalized Sequential Machine (GSM) [1] in which the output alphabet is fixed to be $\{0,1\}$ and all states are considered final, so that its computation on any input produces a binary string.

Formally, a *deterministic finite-state transducer (DFT)* is a 5-tuple $D = (Q, \Sigma, \delta, \lambda, q_1)$, where Q is a finite set of *states*; Σ is a finite *input alphabet*; $\delta: Q \times \Sigma \to Q$ is a *transition function*; $\lambda: Q \times \Sigma \to \{0,1\}^*$ is an *output function*; and $q_1 \in Q$ is the *initial state*. The transition function and output function are extended to $\delta: Q \times \Sigma^* \to Q$ and $\lambda: Q \times \Sigma^* \to \{0,1\}^*$, defined recursively by

$$\begin{aligned} \delta(q,\epsilon) &= q, \\ \delta(q,\sigma x) &= \delta(\delta(q,\sigma),x); \\ \lambda(q,\epsilon) &= \epsilon, \\ \lambda(q,\sigma x) &= \lambda(q,\sigma)\lambda(\delta(q,\sigma),x). \end{aligned}$$

The *output function* $D: \Sigma^* \to \{0,1\}^*$ is defined by $D(x) = \lambda(q_1, x)$ and the *counting function* $\#D: \Sigma^* \to \mathcal{N}$ is defined by $\#D(x) = \#(D(x))$. Intuitively, the counting function $\#D(x)$ is the number represented by the binary string produced by the transduction of D on input x. We extend the output and counting functions to languages $L \subseteq \Sigma^*$ in the usual way. We consider the range of a DFT to be the set of binary strings $s(\#D(\Sigma^*))$. Note that it can be obtained from $D(\Sigma^*)$ by truncating the leading zeroes of each string. The class of counting functions of DFTs is defined by $\#\text{DFT} = \{\, \#D \mid D \text{ is a DFT} \,\}$, and the class of their ranges is defined by $\text{range}(\#\text{DFT}) = \{\, s(\#D(\Sigma^*)) \mid D \text{ is a DFT with input alphabet } \Sigma \,\}$.

3. Inclusions among Counting Functions and their Ranges

In this section, we show that the counting functions computed by deterministic finite-state transducers are properly included among those computed by nondeterministic finite-state automata, and give a counting NFA whose range is not context-free. We denote the class of regular and context-free languages by REG and CFL, respectively. Since the range of every deterministic Generalized Sequential Machine (DGSM) is regular [2], and a DFT is a special case of a DGSM, it follows that the range of a DFT (with leading zeroes truncated) is regular.

Theorem 3.1. range(#DFT) $\subseteq$ REG.

Theorem 3.2. #DFT $\subseteq$ #NFA.

Proof. Let $D = (Q, \Sigma, \delta, \lambda, q_1)$ be a DFT, and construct an NFA M with input alphabet Σ such that for $x \in \Sigma^*$, $\#M(x) = \#D(x)$. Suppose $Q = \{q_1, \ldots, q_s\}$, and let $l = \max\{\, |\lambda(q,\sigma)| \mid q \in Q \wedge \sigma \in \Sigma \,\}$. Let $M = (Q', \Sigma, \delta', \{q_1\}, F)$, where $Q' = Q \cup \{\, q_i^j \mid 1 \le i \le s \wedge 1 \le j \le 2^l \,\}$, $F = \{\, q_i^j \mid 1 \le i \le s \wedge 1 \le j \le 2^l \,\}$, and δ' is defined by

$$\delta'(q_i, \sigma) = \{\delta(q_i,\sigma)\} \cup \{\, \delta(q_i,\sigma)^j \mid 1 \le j \le \#(\lambda(q_i,\sigma)) \,\},$$
$$\delta'(q_i^k, \sigma) = \{\, \delta(q_i,\sigma)^j \mid 1 \le j \le 2^{|\lambda(q_i,\sigma)|} \,\}.$$

First we prove, by induction on $|x|$, that $\#\delta'(q_i^k, x) = 2^{|\lambda(q_i,x)|}$. For the empty string ϵ,

$$\#\delta'(q_i^k, \epsilon) = \begin{cases} 1, & \text{if } q_i^k \in F; \\ 0, & \text{if } q_i^k \notin F \end{cases} = 1 = 2^{|\epsilon|} = 2^{|\lambda(q_i,\epsilon)|},$$

and for strings σx of length at least 1,

$$\begin{aligned} \#\delta'(q_i^k, \sigma x) &= \sum_{p \in \delta'(q_i^k,\sigma)} \#\delta'(p, x) \\ &= \sum_{1 \le j \le 2^{|\lambda(q_i,\sigma)|}} \#\delta'(\delta(q_i,\sigma)^j, x) \\ &= 2^{|\lambda(q_i,\sigma)|} \cdot 2^{|\lambda(\delta(q_i,\sigma),x)|} \\ &= 2^{|\lambda(q_i,\sigma)\lambda(\delta(q_i,\sigma),x)|} \\ &= 2^{|\lambda(q_i,\sigma x)|}. \end{aligned}$$

Next we prove, by induction on $|x|$, the claim (*) $\#\delta'(q_i, x) = \#(\lambda(q_i, x))$. For the empty string ϵ,

$$\#\delta'(q_i, \epsilon) = \begin{cases} 1, & \text{if } q_i \in F; \\ 0, & \text{if } q_i \notin F \end{cases} = 0 = \#(\epsilon) = \#(\lambda(q_i,\epsilon)),$$

and for strings σx of length at least 1,

$$\begin{aligned}\#\delta'(q_i,\sigma x) &= \sum_{p\in\delta'(q_i,\sigma)} \#\delta'(p,x)\\ &= \Big(\sum_{1\le j\le \#(\lambda(q_i,\sigma))} \#\delta'(\delta(q_i,\sigma)^j,x)\Big) + \#\delta'(\delta(q_i,\sigma),x)\\ &= \#(\lambda(q_i,\sigma))\cdot 2^{|\lambda(\delta(q_i,\sigma),x)|} + \#(\lambda(\delta(q_i,\sigma),x))\\ &= \#(\lambda(q_i,\sigma)\lambda(\delta(q_i,\sigma),x))\\ &= \#(\lambda(q_i,\sigma x)).\end{aligned}$$

If we take $i=1$ in $(*)$, then

$$\#M(x) = \#\delta'(q_1,x) = \#(\lambda(q_1,x)) = \#(D(x)) = \#D(x). \quad \blacksquare$$

Lemma 3.3. *#NFA is closed under addition and multiplication.*

Proof idea. (addition) Let $M' = (Q',\Sigma,\delta',I',F')$, $M'' = (Q'',\Sigma,\delta'',I'',F'')$ be NFAs, and construct an NFA M with input alphabet Σ such that for $x\in\Sigma^*$, $\#M(x) = \#M'(x) + \#M''(x)$. M is obtained by a disjoint union construction. The states, transitions, initial states, and final states of M are the disjoint unions of those in M' and M''.

(multiplication) Let $M' = (Q',\Sigma,\delta',I',F')$, $M'' = (Q'',\Sigma,\delta'',I'',F'')$ be NFAs, and construct an NFA M with input alphabet Σ such that for $x\in\Sigma^*$, $\#M(x) = \#M'(x)\cdot\#M''(x)$. M is obtained by a cartesian product construction. The states, transitions, initial states, and final states of M are the cartesian products of those in M' and M''. ■

In this research, we give two examples of counting NFAs whose ranges are not context-free. The first is presented here, and the second—whose range is the binary encodings of the composite numbers—is presented in §4.

Example. *A counting NFA whose range is* $L = \{\,1^n0^n1^n \mid n\in\mathcal{N}\,\}$.

We construct an NFA M with input alphabet $\{0\}$ such that $s(\#M(0^*)) = L$. For $k\in\mathcal{N}$, define an NFA $M_k = (Q,\{0\},\delta,I,F)$, where $Q = \{q_1,\dots,q_{2k},p_1,\dots,p_{2k+1}\}$, $I=\{q_1\}$, $F=\{p_1,\dots,p_{2k+1}\}$, and δ is defined by

$$\begin{aligned}\delta(q_i,0) &= \{q_1,\dots,q_{2k},p_1,\dots,p_{2k}\},\\ \delta(p_i,0) &= \{p_1,\dots,p_{2k+1}\}.\end{aligned}$$

First we prove, by induction on n, that $\#\delta(p_i,0^n) = 2^{(k+1)n}$. For $n=0$,

$$\#\delta(p_i,\epsilon) = \begin{cases}1, & \text{if } p_i\in F;\\ 0, & \text{if } p_i\notin F\end{cases} = 1 = 2^{(k+1)0},$$

and for $n>0$,

$$\begin{aligned}\#\delta(p_i,0^n) &= \sum_{p\in\delta(p_i,0)} \#\delta(p,0^{n-1})\\ &= \sum_{1\le j\le 2^{k+1}} \#\delta(p_j,0^{n-1})\\ &= 2^{k+1}\cdot 2^{(k+1)(n-1)} = 2^{(k+1)n}.\end{aligned}$$

Next we prove, by induction on n, the claim $(*)$ $\#\delta(q_i, 0^n) = 2^{(k+1)n} - 2^{kn}$. For $n = 0$,

$$\#\delta(q_i, \epsilon) = \begin{cases} 1, & \text{if } q_i \in F; \\ 0, & \text{if } q_i \notin F \end{cases} = 0 = 2^{(k+1)0} - 2^{k \cdot 0},$$

and for $n > 0$,

$$\begin{aligned} \#\delta(q_i, 0^n) &= \sum_{p \in \delta(q_i, 0)} \#\delta(p, 0^{n-1}) \\ &= \Big(\sum_{1 \le j \le 2^k} \#\delta(q_j, 0^{n-1}) \Big) + \Big(\sum_{1 \le j \le 2^k} \#\delta(p_j, 0^{n-1}) \Big) \\ &= 2^k \cdot (2^{(k+1)(n-1)} - 2^{k(n-1)}) + 2^k \cdot 2^{(k+1)(n-1)} \\ &= 2^{(k+1)n} - 2^{kn}. \end{aligned}$$

Using Lemma 3.3, construct an NFA M with input alphabet $\{0\}$ such that for $n \in \mathcal{N}$, $\#M(0^n) = \#M_2(0^n) + \#M_0(0^n)$. If we take $i = 1$ in $(*)$, then

$$\begin{aligned} s(\#M(0^*)) &= \{ s(\#M(0^n)) \mid n \in \mathcal{N} \} \\ &= \{ s(\#M_2(0^n) + \#M_0(0^n)) \mid n \in \mathcal{N} \} \\ &= \{ s((2^{3n} - 2^{2n}) + (2^n - 2^0)) \mid n \in \mathcal{N} \} \\ &= \{ s(8^n - 4^n + 2^n - 1) \mid n \in \mathcal{N} \} \\ &= \{ 1^n 0^n 1^n \mid n \in \mathcal{N} \} = L. \quad \blacksquare \end{aligned}$$

Corollary 3.4. range(#NFA) $\not\subseteq$ CFL.

Corollary 3.5. range(#DFT) $\subset$ range(#NFA) *(*range(#DFT) *is properly contained in* range(#NFA)*)*.

Proof. It follows from Theorems 3.1, 3.2, Corollary 3.4, and the fact that REG $\subset$ CFL. ■

4. The Complexity of Counting Functions and their Ranges

In this section, we examine the complexity of computing the counting function of an NFA, and the complexity of recognizing its range. The latter problem—deciding whether a given binary string represents the number of accepting computations on some input—is considered both for a fixed NFA and when the NFA is given as an additional parameter. We show that a fixed counting NFA's range is context-sensitive, and suggest an intractible lower bound by showing that the composite numbers—which are not known to be in P—are the range of a counting NFA. The second of these is called the range membership problem for counting NFAs and is shown to be PSPACE-complete.

An important tool which we use in solving these problems is a matrix algebraic characterization of the counting function of an NFA which allows us to compute it in polynomial time and linear space. Let $M = (Q, \Sigma, \delta, I, F)$ be an NFA with state set $Q = \{q_1, \ldots, q_s\}$, and let $x = \sigma_n \ldots \sigma_1 \in \Sigma^*$. For each $\sigma \in \Sigma$, let $\vec{e}$, A^σ, $\vec{f}$ be the $1\times s$, $s\times s$, $s\times 1$ matrices defined by

$$\vec{e} = \begin{pmatrix} e_1 & e_2 & \ldots & e_s \end{pmatrix}, \quad \text{where } e_j = \begin{cases} 1, & \text{if } q_j \in I; \\ 0, & \text{if } q_j \notin I, \end{cases}$$

$$A^\sigma = \begin{pmatrix} A^\sigma_{11} & A^\sigma_{12} & \cdots & A^\sigma_{1s} \\ A^\sigma_{21} & A^\sigma_{22} & \cdots & A^\sigma_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ A^\sigma_{s1} & A^\sigma_{s2} & \cdots & A^\sigma_{ss} \end{pmatrix}, \quad \text{where } A^\sigma_{ij} = \begin{cases} 1, & \text{if } q_j \in \delta(q_i, \sigma); \\ 0, & \text{if } q_j \notin \delta(q_i, \sigma), \end{cases}$$

$$\vec{f} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \end{pmatrix}, \quad \text{where } f_i = \#\delta(q_i, \epsilon).$$

The symbol $*$ denotes the usual matrix multiplication.

Lemma 4.1. $\vec{e} * A^{\sigma_n} * \cdots * A^{\sigma_1} * \vec{f} = \#M(x)$.

Proof. We prove, by induction on n, the following claim for $1 \le i \le s$:

$$(*) \qquad (A^{\sigma_n} * \cdots * A^{\sigma_1} * \vec{f})_i = \#\delta(q_i, \sigma_n \ldots \sigma_1).$$

For $n = 0$, $(\vec{f})_i = f_i = \#\delta(q_i, \epsilon)$, and for $n > 0$,

$$\begin{aligned} (A^{\sigma_n} * \cdots * A^{\sigma_1} * \vec{f})_i &= \sum_{j=1}^{s} A^{\sigma_n}_{ij} \cdot (A^{\sigma_{n-1}} * \cdots * A^{\sigma_1} * \vec{f})_j \\ &= \sum_{j=1}^{s} A^{\sigma_n}_{ij} \cdot \#\delta(q_j, \sigma_{n-1} \ldots \sigma_1) \\ &= \sum_{q_j \in \delta(q_i, \sigma_n)} \#\delta(q_j, \sigma_{n-1} \ldots \sigma_1) \\ &= \#\delta(q_i, \sigma_n \ldots \sigma_1). \end{aligned}$$

Applying $(*)$, we have

$$\begin{aligned} \vec{e} * A^{\sigma_n} * \cdots * A^{\sigma_1} * \vec{f} &= \sum_{j=1}^{s} \vec{e}_j \cdot (A^{\sigma_n} * \cdots * A^{\sigma_1} * \vec{f})_j \\ &= \sum_{j=1}^{s} \vec{e}_j \cdot \#\delta(q_j, x) \\ &= \sum_{q_j \in I} \#\delta(q_j, x) \\ &= \#M(x). \quad \blacksquare \end{aligned}$$

The algebraic characterization of Lemma 4.1 gives us the following algorithm for computing $\#M(x)$ which processes the symbols of x from right to left, producing an s-entry column vector after each of n matrix multiplications.

```
input $x$;   $\{= \sigma_n \ldots \sigma_1 \in \Sigma^*\}$
$\vec{v} := \vec{f}$;
for $i := 1$ to $n$ do
     $\vec{v} := A^{\sigma_i} * \vec{v}$;
output $\vec{e} * \vec{v}$
```

After n multiplications, we obtain $A^{\sigma_n} * \cdots * A^{\sigma_1} * \vec{f}$, whose ith entry is $\#\delta(q_i, x)$. This computation can be done in time polynomial in n and, since $\#\delta(q_i, x) \leq s^n$, each entry can be represented in binary using space linear in n.

In the remainder of this section, we turn our attention to the ranges of counting NFAs. We apply the method of computing $\#M(x)$ given by the previous algorithm to show that the range $s(\#M(\Sigma^*))$ of a counting NFA M is context-sensitive, i.e., is in NSPACE(n).

Given a binary string y, how can we decide if $y \in s(\#M(\Sigma^*))$? That is, how can we decide if y represents the number of accepting computations of M on some input x? A first approach using Lemma 4.1 is to guess symbols $\sigma_1, \ldots, \sigma_n$ of x from right to left, computing after each guess a column vector $\vec{v}$ whose ith entry is $v_i = \#\delta(q_i, x)$, and accepting if and only if y is the binary representation of $\vec{e} * \vec{v} = \#M(x)$:

```
input $y$;
$\vec{v} := \vec{f}$;
while true do
     begin
          if $s(\vec{e} * \vec{v}) = y$ then accept;
          guess $\sigma \in \Sigma$;
          $\vec{v} := A^{\sigma} * \vec{v}$
     end
```

Some computations of this nondeterministic algorithm may not halt and will require an unbounded amount of space in which to store the entries of $\vec{v}$. In the following development, we show how to impose a linear space bound on the computations of this algorithm by placing a cap on the size of the entries of $\vec{v}$.

Let $y \in \{0,1\}^*$. We define $\text{cap}_y \colon \mathcal{N} \to \mathcal{N}$ by $\text{cap}_y(m) = \min\{m, \#(y) + 1\}$. We extend cap_y to matrices of natural numbers by applying cap_y to each entry of the matrix. We will need the following properties of the cap function in order to impose a bound on the space required by the previous algorithm and maintain its correctness.

Lemma 4.2. *Let $y \in \{0,1\}^*$; $m, l \in \mathcal{N}$; and A, B compatible matrices of natural numbers.*

(1) $s(\text{cap}_y(m)) = y \iff s(m) = y$

(2) $\text{cap}_y(m + l) = \text{cap}_y(\text{cap}_y(m) + \text{cap}_y(l))$

(3) $\text{cap}_y(m \cdot l) = \text{cap}_y(m \cdot \text{cap}_y(l))$

(4) $\text{cap}_y(A * B) = \text{cap}_y(A * \text{cap}_y(B))$

Proof. (1) $s(\mathrm{cap}_y(m)) = y \iff s(\min\{m, \#(y)+1\}) = y \iff s(m) = y$.
(2) We consider two cases. If $m + l > \#(y)$, then $\mathrm{cap}_y(m+l) = \mathrm{cap}_y(\mathrm{cap}_y(m) + \mathrm{cap}_y(l)) = \#(y)+1$. If $m+l \le \#(y)$, then $\mathrm{cap}_y(m+l) = \mathrm{cap}_y(\mathrm{cap}_y(m)+\mathrm{cap}_y(l)) = m+l$.
(3) Proof is similar to (2).
(4)

$$\begin{aligned}
\mathrm{cap}_y(A * B)_{ik} &= \mathrm{cap}_y\Big(\sum_j A_{ij} \cdot B_{jk}\Big)\\
&= \mathrm{cap}_y\Big(\sum_j \mathrm{cap}_y(A_{ij} \cdot B_{jk})\Big), \quad \text{by (2);}\\
&= \mathrm{cap}_y\Big(\sum_j \mathrm{cap}_y(A_{ij} \cdot \mathrm{cap}_y(B_{jk}))\Big), \quad \text{by (3);}\\
&= \mathrm{cap}_y\Big(\sum_j A_{ij} \cdot \mathrm{cap}_y(B_{jk})\Big), \quad \text{by (2);}\\
&= \mathrm{cap}_y(A * \mathrm{cap}_y(B))_{ik}. \quad \blacksquare
\end{aligned}$$

Theorem 4.3. $\mathrm{range}(\#\mathrm{NFA}) \subseteq \mathrm{CSL}$.

Proof. We show that for a fixed NFA M with input alphabet Σ, $s(\#M(\Sigma^*))$ is in NSPACE(n). Consider the following modification of our previous algorithm which decides whether or not $y \in s(\#M(\Sigma^*))$:

```
input $y$;
$\vec{v} := \vec{f}$;
while true do
    begin
        if $s(\mathrm{cap}_y(\vec{e} * \vec{v})) = y$ then accept;
        guess $\sigma \in \Sigma$;
        $\vec{v} := \mathrm{cap}_y(A^\sigma * \vec{v})$
    end
```

The matrices $\vec{e}$, A^σ, and $\vec{f}$ can be kept in finite control and the space required by $\vec{v}$ is $O(|y|)$, since its entries are at most $\#(y)+1$; therefore, this algorithm can be implemented by a nondeterministic linear space-bounded Turing machine. To show correctness, let $\sigma_1, \ldots, \sigma_n$ be a sequence of guesses of the algorithm and $x = \sigma_n \ldots \sigma_1$. By Lemma 4.2(4), the value of $\vec{v}$ at the beginning of the while-loop after guessing x will be

$$\begin{aligned}
\vec{v} &= \mathrm{cap}_y(A^{\sigma_n} * \mathrm{cap}_y(A^{\sigma_{n-1}} * \cdots * \mathrm{cap}_y(A^{\sigma_1} * \vec{f}) \cdots))\\
&= \mathrm{cap}_y(A^{\sigma_n} * \cdots * A^{\sigma_1} * \vec{f}).
\end{aligned}$$

By this observation, Lemma 4.1, and Lemma 4.2(1,4), we have

$$\begin{aligned}
s(\mathrm{cap}_y(\vec{e} * \vec{v})) = y &\iff s(\mathrm{cap}_y(\vec{e} * \mathrm{cap}_y(A^{\sigma_n} * \cdots * A^{\sigma_1} * \vec{f}))) = y\\
&\iff s(\mathrm{cap}_y(\vec{e} * A^{\sigma_n} * \cdots * A^{\sigma_1} * \vec{f})) = y\\
&\iff s(\mathrm{cap}_y(\#M(x))) = y\\
&\iff s(\#M(x)) = y,
\end{aligned}$$

so the algorithm accepts y if and only if $s(\#M(x)) = y$, for some $x \in \Sigma^*$. ∎

It is interesting to consider whether the information in this algorithm can be further compressed into space which is logarithmic in $|y|$, giving us an NSPACE$(\log(|y|))$·algorithm for recognizing the range of a counting NFA. In the following example, we give evidence that, if possible, it will be difficult to achieve, by showing that the composite numbers—which are not known to be in P—are the range of a counting NFA.

Example. *A counting NFA whose range is Composites* $\cup \{0\}$.

We construct an NFA M with input alphabet Σ such that $\#M(\Sigma^*) = \text{Composites} \cup \{0\}$. First, construct an NFA M' with input alphabet $\Sigma = \{0,1\}$ such that $\#M'(0^m10^l) = m \cdot l$. Let M' be the NFA pictured in the transition graph of Figure 4.1.

$$\begin{tikzcd}
\arrow[r] & q_1 \arrow[loop below, "0"] \arrow[rr, bend left, "0"] & & q_2 \arrow[loop below, "0"] \arrow[ll, bend left, "1"]
\end{tikzcd}$$

Figure 4.1. A counting NFA which multiplies unary numbers

We prove, by induction on l, the following claims:

$$\#\delta(q_1, 0^l) = l;$$
$$\#\delta(q_2, 0^l) = 1.$$

For $l = 0$,

$$\#\delta(q_1, \epsilon) = \begin{cases} 1, & \text{if } q_1 \in F; \\ 0, & \text{if } q_1 \notin F \end{cases} = 0;$$
$$\#\delta(q_2, \epsilon) = \begin{cases} 1, & \text{if } q_2 \in F; \\ 0, & \text{if } q_2 \notin F \end{cases} = 1,$$

and for $l > 0$,

$$\begin{aligned}
\#\delta(q_1, 0^l) &= \sum_{p \in \delta(q_1, 0)} \#\delta(p, 0^{l-1}) \\
&= \#\delta(q_1, 0^{l-1}) + \#\delta(q_2, 0^{l-1}) \\
&= (l-1) + 1 = l; \\
\#\delta(q_2, 0^l) &= \sum_{p \in \delta(q_2, 0)} \#\delta(p, 0^{l-1}) \\
&= \#\delta(q_2, 0^{l-1}) = 1.
\end{aligned}$$

Next we prove, by induction on m, the following claims:

$$\begin{aligned}\#\delta(q_1, 0^m10^l) &= m \cdot l;\\ \#\delta(q_2, 0^m10^l) &= l.\end{aligned}$$

For $m = 0$,

$$\begin{aligned}\#\delta(q_1, 10^l) &= \sum_{p\in\delta(q_1,1)} \#\delta(p, 0^l) = 0 = 0 \cdot l;\\ \#\delta(q_2, 10^l) &= \sum_{p\in\delta(q_2,1)} \#\delta(p, 0^l)\\ &= \#\delta(q_1, 0^l) = l,\end{aligned}$$

and for $m > 0$,

$$\begin{aligned}\#\delta(q_1, 0^m10^l) &= \sum_{p\in\delta(q_1,0)} \#\delta(p, 0^{m-1}10^l)\\ &= \#\delta(q_1, 0^{m-1}10^l) + \#\delta(q_2, 0^{m-1}10^l)\\ &= (m-1) \cdot l + l = m \cdot l;\\ \#\delta(q_2, 0^m10^l) &= \sum_{p\in\delta(q_2,0)} \#\delta(p, 0^{m-1}10^l)\\ &= \#\delta(q_2, 0^{m-1}10^l) = l.\end{aligned}$$

Therefore, we have $\#M'(0^m10^l) = \#\delta(q_1, 0^m10^l) = m \cdot l$. Consider the regular language $R = \{\, 0^m10^l \mid m, l \geq 2 \,\}$. Let M'' be a DFA which accepts R. Then

$$\#M''(0^m10^l) = \begin{cases} 1, & \text{if } m, l \geq 2;\\ 0, & \text{otherwise.}\end{cases}$$

Using Lemma 3.3, construct an NFA M such that $\#M(x) = \#M'(x) \cdot \#M''(x)$.

$$\begin{aligned}\#M(0^m10^l) &= \#M'(0^m10^l) \cdot \#M''(0^m10^l)\\ &= \begin{cases} m \cdot l, & \text{if } m, l \geq 2;\\ 0, & \text{otherwise,}\end{cases}\end{aligned}$$

so $\#M(\Sigma^*) = \text{Composites} \cup \{0\}$. ∎

When the range membership problem is considered as a function of both a given NFA and binary string, we are able to pinpoint its complexity by giving a completeness result for PSPACE.

Range Membership

Instance: M, an NFA with input alphabet Σ; $y \in \{0, 1\}^*$.

Question: $y \in s(\#M(\Sigma^*))$?

Theorem 4.4. *Range Membership is PSPACE-complete.*

Proof. We have shown in Theorem 4.3 that membership can be decided nondeterministically using space $O(\|Q\| \cdot |y|)$, where $\|Q\|$ is the number of states in M. By Savitch's Theorem, Range Membership $\in$ NSPACE(n^2) $\subseteq$ PSPACE. We show hardness by logspace reduction from the nonuniversality problem for NFAs, which was proved PSPACE-complete by Stockmeyer and Meyer [5,6]. Let M be an NFA with input alphabet Σ.

$$\begin{aligned} L(M) \neq \Sigma^* &\iff \exists x \in \Sigma^*,\ x \notin L(M) \\ &\iff \exists x \in \Sigma^*,\ \#M(x) = 0 \\ &\iff 0 \in \#M(\Sigma^*) \\ &\iff \epsilon \in s(\#M(\Sigma^*)). \quad \blacksquare \end{aligned}$$

5. Pumping Behavior and Linear Recurrences

In this section, we consider the pumping behavior of a counting finite-state automaton. For a fixed input string, we show that the number of accepting computations—considered as a function of the number of times a fixed substring of the input is pumped—satisfies a homogeneous linear recurrence equation of finite degree having integer coefficients. We precede this result with some relevant definitions and facts from the theories of recurrence equations and matrices.

Let $g\colon \mathcal{N} \to \mathcal{N}$. *$g$ satisfies a homogeneous linear recurrence equation of degree s having integer coefficients* if there exist $a_1, \ldots, a_s \in Z$ such that for $n \in \mathcal{N}$,

$$g(n+s) = \sum_{k=1}^{s} a_k \cdot g(n+s-k).$$

Let A be an $s \times s$ matrix of integers, and I be the $s \times s$ identity matrix with 1's on the diagonal and 0's elsewhere. The *characteristic polynomial* of A is the polynomial p defined by $p(\lambda) = \det(A - \lambda \cdot I)$, where det is the determinant function. Note that the characteristic polynomial is of degree s and has integer coefficients, since A has integer entries. The *characteristic equation* of A is the equation $\det(A - \lambda \cdot I) = 0$. The characteristic polynomial is said to be *monic*, since the coefficient of λ^s is $(-1)^s = \pm 1$; therefore, the characteristic equation can be written as

$$\lambda^s = \sum_{k=1}^{s} a_k \cdot \lambda^{s-k},$$

where $a_1, \ldots, a_s \in Z$. One of the most important results in matrix theory is the Cayley-Hamilton Theorem, which states that a matrix satisfies its own characteristic equation. We use it to analyze the pumping behavior of a counting finite-state automaton.

Theorem 5.1. *Let $M = (Q, \Sigma, \delta, I, F)$ be an NFA with state set $Q = \{q_1, \ldots, q_s\}$, and let $w, x, z \in \Sigma^*$. There exist $a_1, \ldots, a_s \in \mathcal{Z}$ such that for every $n \in \mathcal{N}$,*

$$\#M(wx^{n+s}z) = \sum_{k=1}^{s} a_k \cdot \#M(wx^{n+s-k}z).$$

Proof. Let $\vec{e}$, A^σ, $\vec{f}$ be defined as in section 4, and let $A^x = A^{\sigma_1} * \cdots * A^{\sigma_{|x|}}$, where $x = \sigma_1 \ldots \sigma_{|x|}$. We define A^w and A^z similarly. As discussed before, the characteristic equation of A^x can be written as

$$\lambda^s = \sum_{k=1}^{s} a_k \cdot \lambda^{s-k},$$

where $a_1, \ldots, a_s \in \mathcal{Z}$. By the Cayley-Hamilton Theorem,

$$A^{x^s} = \sum_{k=1}^{s} a_k \cdot A^{x^{s-k}}.$$

By this observation and Lemma 4.1, we have for $1 \le i \le s$ and $n \in \mathcal{N}$,

$$\begin{aligned}
\#\delta(q_i, wx^{n+s}z) &= (A^w * A^{x^{n+s}} * A^z * \vec{f})_i \\
&= (A^w * A^{x^n} * A^{x^s} * A^z * \vec{f})_i \\
&= \left(A^w * A^{x^n} * \left(\sum_{k=1}^{s} a_k \cdot A^{x^{s-k}}\right) * A^z * \vec{f}\right)_i \\
&= \left(\sum_{k=1}^{s} a_k \cdot \left(A^w * A^{x^n} * A^{x^{s-k}} * A^z * \vec{f}\right)\right)_i \\
&= \sum_{k=1}^{s} a_k \cdot (A^w * A^{x^{n+s-k}} * A^z * \vec{f})_i \\
&= \sum_{k=1}^{s} a_k \cdot \#\delta(q_i, wx^{n+s-k}z).
\end{aligned}$$

$$\begin{aligned}
\#M(wx^{n+s}z) &= \sum_{q_i \in I} \#\delta(q_i, wx^{n+s}z) \\
&= \sum_{q_i \in I} \left(\sum_{k=1}^{s} a_k \cdot \#\delta(q_i, wx^{n+s-k}z)\right) \\
&= \sum_{k=1}^{s} a_k \cdot \left(\sum_{q_i \in I} \#\delta(q_i, wx^{n+s-k}z)\right) \\
&= \sum_{k=1}^{s} a_k \cdot \#M(wx^{n+s-k}z). \quad \blacksquare
\end{aligned}$$

Stearns and Hunt [4] showed that the number of accepting computations over all inputs of a given length—considered as a function of the length—satisfies a homogeneous linear recurrence equation of finite degree having rational coefficients.

The technique of Theorem 5.1 can be used to strengthen and simplify the proof of their result, obtaining integer coefficients and a recurrence equation which is satisfied regardless of which state is considered to be the start state, by applying the Cayley-Hamilton Theorem with the matrix $A = \sum_{\sigma \in \Sigma} A^{\sigma}$.

6. Summary and Open Questions

We summarize some of the results contained heretofore, and ask open questions about improvements and extensions of our results.

In §3, we showed that range(#NFA) includes range(#DFT)—the ranges of deterministic finite-state transducers. It follows from the Generalized Sequential Machine results of Ginsburg and Greibach [2] that the latter is the class of all regular languages comprised of binary strings without leading zeroes. Is the class of all context-free languages comprised of binary strings without leading zeroes included in range(#NFA)?

In §4, we showed that the range $s(\#M(\Sigma^*))$ of a counting NFA M is in NSPACE(n). How tight is this upper bound? Respecting the fact that the composite numbers are the range of a counting NFA, is there a subclass of NSPACE(n) which contains range(#NFA)? Are there ranges of counting NFAs which are complete for NSPACE(n)? NP? some other time- or space-bounded complexity class?

In §5, we showed that for a fixed input string, the number of accepting computations—considered as a function of the number of times a fixed substring of the input is pumped—satisfies a homogeneous linear recurrence equation of finite degree having integer coefficients. Does this lead to a simple pumping lemma which can be used to show that a function is not in #NFA or a language is not in range(#NFA)? Which arbitrary functions satisfying linear recurrences as in Theorem 5.1 are computed by counting NFAs? That is, can we precisely characterize #NFA as a class of functions satisfying a restricted class of recurrence equations?

References

[1] Ginsburg, S. "Examples of abstract machines," *IEEE Trans. on Electronic Computers* **11**: 2 (1962), 132–135.

[2] Ginsburg, S., and Greibach, S.A. "Abstract families of languages," *Studies in Abstract Families of Languages*, pp. 1–32, Memoir No. 87, American Mathematical Society, Providence, R.I., 1969.

[3] Hopcroft, J.E., and Ullman, J.D. "Introduction to automata theory, languages, and computation," Addison-Wesley, Reading, Mass., 1979.

[4] Stearns, R.E., and Hunt, H.B. III. "On the equivalence and containment problems for unambiguous regular expressions, regular grammars, and finite automata," *SIAM J. Comput.* **14** (1985), 598–611.

[5] Stockmeyer, L.J., and Meyer, A.R. "Word problems requiring exponential time," *Proc. Fifth Annual ACM Symposium on the Theory of Computing* (1973), 1–9.

[6] Stockmeyer, L.J. "The Complexity of Decision Problems in Automata Theory and Logic," Doctoral Thesis, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Mass., 1974.

[7] Valiant, L.G. "The complexity of computing the permanent," *Theor. Comput. Sci.* 8 (1979), 189–201.

[8] Valiant, L.G. "The complexity of enumeration and reliablity problems," *SIAM J. Comput.* 8: 3 (1979), 410–421.

(Received July 29, 1988)

Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1990

Decision problems arising from knapsack transformations

Arto Salomaa*

Abstract

The transformations of knapsack vectors by modular multiplications give rise to various intriguing decidability questions. While the most important applications of the resulting algorithms belong to cryptanalysis, the algorithms are certainly of interest on their own right. The basic problem we are considering is whether or not a given vector is obtained from a super-increasing vector by one or several modular multiplications. Various formulations of this problem, as well as various other problems connected with it, will also be discussed. Some interesting problems remain open.

1 Introduction. Connection with cryptography

It is well known that the *knapsack problem* is NP-complete. It is an especially lucid example of an NP- complete problem - easily explainable even for a layman. We are given a vector $B = (b_1, \ldots, b_n)$ consisting of distinct positive integers, as well as another positive integer α. If possible, we have to find a subset of the set $\{b_1, \ldots, b_n\}$ such that the elements in this subset sum up to α. Equivalently, we have to find a column vector C consisting of 0's and 1's such that $BC = \alpha$.

Knapsack vectors B can be used to encrypt messages as follows. A message is divided into blocks C consisting of n bits. Such a block is encrypted as the number BC. Even if one knows the vector B, decryption still amounts to solving the NP-complete knapsack problem. However, decryption is equally difficult for the legal recipient of the message and an illegal eavesdropper, usually called a cryptanalyst. To obtain a public-key cryptosystem, the legal recipient must be given some secret trapdoor information about the publicized vector B. In the earliest public-key cryptosystem, [1], this is done in terms of super-increasing vectors.

A vector $A = (a_1, \ldots, a_n)$ is termed *super-increasing* iff each entry in A exceeds the sum of the preceding entries. The resulting knapsack problems (A, α) are easy and can be solved as follows by just scanning A once from right to left. Thus, we want to determine a bit vector $C = (c_1, \ldots, c_n)^T$ such that $AC = \alpha$. Clearly, $c_n = 1$ iff $\alpha \geq a_n$. (If we do not include a_n in the sum, we cannot reach α because $\sum_{i=1}^{n-1} a_i < a_n$.) Next we compare a_{n-1} with $\alpha - a_n$ or α, depending whether $c_n = 1$ or $c_n = 0$. And so forth, until we reach a_1. As a consequence we observe that every knapsack problem (A, α), where A is super-increasing, possesses at most

*Mathematics Department, University of Turku, SF-20500 Turku, Finland one solution. This is a property of knapsack vectors referred to as injectivity in the sequel.

A super-increasing knapsack vector A can be scrambled by *modular multiplications*. One chooses a multiplier t and modulus m and reduces, for each i, the product ta_i modulo m. The resulting vector $B = (b_1, \ldots, b_n)$ is publicized. For technical reasons, it is assumed that the greatest common division $(t, m) = 1$ and that $m > \sum_{i=1}^{n} a_i$. Consequently, t possesses an inverse $t^{-1} = u \pmod{m}$. The pair (t, m) constitutes the trapdoor information known to the legal recipient who can form the superincreasing A and decrypt a cryptotext β by using A and the smallest positive remainder α of $u\beta$ modulo m. More details and examples can be found in [2] and [3].

A cryptanalyst has to solve the knapsack problem determined by A' and α' that looks like an arbitrary knapsack problem. However, it only looks like it because very few knapsack vectors are reachable by a modular multiplication from a superincreasing vector. Indeed, Shamir, [5], gave an algorithm working in random polynomial time for finding a super-increasing vector A' (not necessarily the same as the original A) such that the given B results from A' by a modular multiplication. A deterministic polynomial-time algorithm, based on different considerations, was given in [4].

The present paper discusses decision problems and algorithms arising in this set-up. In what follows, the interconnections with cryptography will not any more be important. All definitions and results will be given in terms of ordered n-tuples of positive integers. The exposition is self-contained. A brief outline of the contents of the paper follows.

In Section 2, the basic definitions and notations are given. They include also concepts needed for our technical apparatus. Section 2 contains also some basic results. Our technical tools will be developed in Sections 3 and 4 which include also illustrative examples.

By definition, a vector B is *super-reachable* iff B results from *some* super-increasing vector by a modular multiplication. The two algorithms given in Section 5 decide whether or not the given vector is super-reachable and, in the positive case, produce the corresponding multiplier and modulus. Consequences, related decision problems and modifications are discussed in Section 6. Section 7 deals with a variant, where also permutations of given vectors are taken into account.

A vector is *hyper-reachable* iff it results from some super-increasing vector by a *sequence* of modular multiplications. The wellknown example presented in the basic paper [1] starts with the super-increasing vector $A = (5,10,20)$. Using the multiplier 17 and modulus 47, we obtain $A' = (38,29,11)$. Another multiplier 3 and modulus 89 are applied to A', yielding $B = (25,87,33)$. Thus, B is hyper-reachable. It is easy to show that B cannot be reached from A by one modular multiplication. However, B is still super-reachable. It results, for instance, from (2,3,66) using the multiplier 62 and modulus 99. (This result is obtained by the initial part of the algorithm presented in Section 5.)

It is shown in Section 5 that there are properly hyper-reachable vectors, that is, hyper-reachable vectors which are not super-reachable. Two algorithms for hyper-reachability with a bounded number of modular multiplications are given in Section 8. Section 8 also discusses some other decision problems dealing with hyper-reachability. The concluding Section 9 contains some open problems.

2 Definitions and notations

An ordered n-tuple of distinct positive integers $A = (a_1, \dots, a_n), n \geq 3$, is referred to as a *knapsack vector* of dimension n. A knapsack vector A is *increasing* (resp. *super-increasing*) iff

$$a_j > a_{j-1} \quad (\text{resp.} \quad a_j > \sum_{i=1}^{j-1} a_i)$$

holds for all $j = 2, \dots, n$. Clearly, every super-increasing vector is inreasing. For a knapsack vector A, we define

$$\max A = \max\{a_j \mid 1 \leq j \leq n\}.$$

For a positive number x, we denote by $[x]$ the integer part of x, that is, the greatest integer $\leq x$. For integers x and $m \geq 2$, we denote by $(x, \bmod m)$ the *least nonnegative remainder* of x modulo m. Clearly,

$$(x, \bmod m) = x - [x/m] \cdot m$$

This equation will be often written in the form

$$x = (x, \bmod m) + [x/m] \cdot m. \tag{1}$$

We now define two different notions of modular multiplication. Consider a knapsack vector A, an integer $m > \max A$ and a positive integer $t < m$ such that the greatest common divisor $(t, m) = 1$. If $B = (b_1, \dots, b_n)$ is a vector such that

$$b_i = (ta_i, \bmod m), \text{ for } i = 1, \dots, n,$$

we write

$$A \xrightarrow[(t,m)]{} B. \tag{2}$$

The integers t and m are referred to as the *multiplier* and the *modulus*, respectively.

Since every knapsack vector A satisfies max $A \geq 3$, we have always $m \geq 4$. The condition $t < m$ is no loss of generality because if $t > m$, we can subtract $[t/m] \cdot m$ from t without affecting the result of modular multiplication. The equation $t = m$ is not possible because $(t, m) = 1$. The latter condition guarantees also that t possesses an inverse $t^{-1} = u$ such that

$$tu \equiv 1 \pmod m$$

and $1 \leq u < m$. Since clearly $m > \max B$, we have also

$$B \xrightarrow[(u,m)]{} A. \tag{3}$$

We now come to the other notion of modular multiplication. It means simply the strengthening of our previous notion, where we make the additional assumption that $m > \sum_{i=1}^{n} a_i$. If this condition is satisfied and (2) holds, we write

$$A \xrightarrow[(M,t,m)]{} B. \tag{4}$$

Observe that now a condition analogous to (3) does not necessarily hold because we cannot conclude that $m > \sum_{i=1}^{n} b_i$. Clearly, (4) implies (2) but not vice versa.

The vector B is (A,t,m)-*reachable* (resp. (A,t,m)-*super-reachable*) iff (2) holds and A is increasing (resp. (4) holds and A is super-increasing). B is *super-reachable* iff B is (A,t,m)-super-reachable, for some triple (A,t,m).

Observe that a notion of reachability, defined analogously to that of super-reachability, does not make much sense because apparently every vector would be reachable.

Let $r \geq 1$ be an integer. A knapsack-vector B is *r-hyper-reachable* iff there is a sequence of vectors $A_0, A_1, \ldots, A_r = B$ such that A_0 is super-increasing and, for each $i = 0, \ldots, r-1$, there are t, m such that

$$A_i \underset{(M,t,m)}{\longrightarrow} A_{i+1}.$$

Observe that t and m may be different for different values of i. The notions of 1-hyper-reachability and super-reachability coincide.

In the Introduction the vector $B = (25,87,33)$ was defined in way which showed that it is 2-hyper-reachable. It was also observed that B is, in fact, super-reachable. Similarly, the derivation chains

$$(1,2,4) \underset{(M,5,8)}{\longrightarrow} (5,2,4) \underset{(M,5,12)}{\longrightarrow} (1,10,8) = B_1$$

and

$$(2,4,7) \underset{(M,7,24)}{\longrightarrow} (14,4,1) \underset{(M,3,20)}{\longrightarrow} (2,12,3) \underset{(M,2,23)}{\longrightarrow} (4,1,6) = B_2$$

show that the vectors B_1 and B_2 are 2-hyper-reachable and 3-hyper-reachable, respectively. It will be seen in Example 2 of Section 5 that B_2 is super-reachable; whereas B_1 is not super-reachable.

We would like to emphasize that all our examples only illustrate some points in the theory and are, therefore, "small". Cryptographically interesting vectors would have to be much bigger, say $n = 200$.

A vector is *hyper-reachable* iff it is r-hyper-reachable, for some r. We now define a notion that enables us to construct easily examples of non-hyper-reachable vectors.

A knapsack vector $A = (a_1, \ldots, a_n)$ is *injective* iff the function $f(C) = AC$, defined for n-dimensional bit column vectors C, is injective. Equivalently, the injectivity of A means that, for every α, the knapsack problem determined by A and α possesses at most one solution.

Theorem 1 *Every hyper-reachable vector is injective. Hence, every super-reachable vector is injective.*

Proof. The theorem is a consequence of the following two facts (i) and (ii).

(i) Every super-increasing vector A is injective. Indeed, assume that $AC = AC'$ holds for some bit column vectors C and C'. Then the last components of C and C' coincide, because if one of c_n and c'_n equals 0 and the other equals 1, the numbers AC and AC' cannot be the same. Similarly we conclude by descending induction that, for all $i = 1, \ldots, n$, $c_i = c'_i$. Consequently, $C = C'$.

(ii) The relation (4) preserves injectivity. Assume the contrary: A is injective and there are two distinct vectors C and C' such that $BC = BC'$. We obtain by (4)

$$B \underset{(u,m)}{\longrightarrow} A,$$

where u is the inverse of t modulo m. This implies that $uBC = uBC'$ and, consequently, $AC \equiv AC' \pmod{m}$. Since $m > \sum_{i=1}^{n} a_i$, we conclude that $AC = AC'$, contradicting the injectivity of A. □

Theorem 1 shows that if a knapsack vector is not injective, it cannot be hyper-reachable. For instance, every vector, where some component equals the sum of some other components, is noninjective. We now come to notions that are quite essential in the subsequent proofs.

Consider a knapsack vector $A = (a_1, \ldots, a_n)$, an integer $m > \max A$ and positive integer $t < m$ such that $(t, m) = 1$. The *growing sequence associated with the triple* (A, t, m) is the sequence of triples $(A(k), t, m + kt)$, $k = 0, 1, 2, \ldots$, where

$$A(k) = (a_1 + k \cdot [ta_1/m], \ldots, a_n + k \cdot [ta_n/m]).$$

Thus, the growing sequence associated with (A, t, m) begins with (A, t, m). The terms *multiplier* and *modulus* refer also to the numbers t and $m + kt$ in the triple $(A(k), t, m + kt)$.

For instance, if $A = (1, 2, 3), t = 4, m = 5$, then the growing sequence begins with the triples

$$((1, 2, 3), 4, 5), ((1, 3, 5), 4, 9) \text{ and } ((1, 4, 7), 4, 13).$$

If $A = (1, 4, 7), t = 3, m = 8$, then the growing sequence is

$$((1, 4 + k, 7 + 2k), 3, 8 + 3k), k = 0, 1, 2, \ldots.$$

As the third example, take $A = (1, 5, 6), t = 7, m = 8$. Then the associated growing sequence is

$$((1, 5 + 4k, 6 + 5k), 7, 8 + 7k), k = 0, 1, 2, \ldots.$$

A number $i, 2 \le i \le n$, is termed a *violation point* in a knapsack vector A iff $a_i \le \sum_{j=1}^{i-1} a_j$. Thus, the ith componenet of A violates the requirement of A being super-increasing. If A is increasing, every violation point i in A satisfies $i \ge 3$.

The *goal* of a triple (A, t, m) (defined as above) is the first triple $(A(k), t, m+kt)$ in the growing sequence such that $A(k)$ is super-increasing and $m + kt$ is greater than the sum of the components of $A(k)$, provided such triples exist. Clearly, a triple can be its own goal and some triples have no goal. In particular, if A is not increasing, then (A, t, m) cannot possess a goal. This follows because $a_i > a_{i+1}$ implies that $[ta_i/m] \ge [ta_{i+1}/m]$ and consequently, for all k,

$$a_i + k \cdot [ta_i/m] > a_{i+1} + k \cdot [ta_{i+1}/m].$$

Returning to the three examples considered above, $i = 3$ is a violation point in the initial vector of the first and third example. In the second example the initial vector, as well as all vectors in the growing sequence are super-increasing. The goal in the first example is the third triple in the growing sequence, although in the first triple neither the vector is super-increasing nor the modulus big enough. The sequences in the second and third examples possess no goals. In the second example the modulus will never become big enough. The same holds true for the third example, although the violation point $i = 3$ in the initial vector is "rescued" already by the second vector (1,9,11). (The formal definition of a rescuer will be given below.)

The following more general result concerning super-reachable vectors can be established already at this stage.

Theorem 2 *The vector $(i, i-1, i-2, \ldots, i-j), i-j \ge 1$, is super-reachable exactly in case if both $j = 2$ and $i \ge 4$.*

Proof. If $j < 2$, the vector is not at all a knapsack vector. If $j > 2$, then the vector is not injective and hence, by Theorem 1, cannot be super-reachable. The same conclusion can be made if $j = 2$ and $i = 3$, because (3,2,1) is not injective. If $j = 2$ and $i > 4$, we have

$$(1,3,5) \xrightarrow[(M,i,2i+1)]{} (i, i-1, i-2). \tag{5}$$

If $j = 2$ and $i = 4$, we have

$$(1,4,7) \xrightarrow[(M,4,13)]{} (4,3,2).$$

In this case (5) does not hold because the modulus is too small but we may use the second triple in the growing sequence. □

We define finally a notion in some sense dual to that of a growing sequence. Let A, B, t, m be such that (2) is satisfied, $t \leq \max B$ and $m > 2\max B$. Then the *diminishing sequence associated with the triple* (A, t, m) is the sequence of triples

$$(A(-k), t, m-kt), 0 \leq k \leq s,$$

where s is the smallest integer such that $m - st \leq 2 \max B$ and the vectors $A(-k)$ are defined by descending induction as follows. $A(-0) = A$. Assume that $A(k) = (d_1, \ldots, d_n)$ has been defined, and that we still have $m - kt > 2 \max B$. (By the choice of m, this condition holds for $k = 0$.) Then

$$A(-k-1) = (d_1 - [td_1/(m-kt)], \ldots, d_n - [td_n/(m-kt)]).$$

Observe that $t < m - st$. Diminishing sequences are always finite, whereas growing sequences are infinite. However, in the sequel only finite initial segments of growing sequences will be of interest.

3 Fundamental lemmas I

We will now develop the technical tools needed. As will be seen, most of the technical apparatus deals with growing and diminishing sequences. We begin with properties of growing sequences. In Lemmas 1-3, the notation A, t, m and $A(k)$ is the same as in the definition of a growing sequence.

Lemma 1 *If A is increasing or super-increasing, then each vector in the growing sequence associated with (A, t, m) is increasing or super-increasing, respectively.*

Proof. The inequality $a_{i-1} < a_i$ implies the inequality $[ta_{i-1}/m] \leq [ta_i/m]$. Hence, if A is increasing then so is every $A(k)$.

Assume, next, that

$$\sum_{j=1}^{i-1} a_j < a_i.$$

Consequently,

$$\sum_{j=1}^{i-1} [ta_j/m] \leq [(t\sum_{j=1}^{i-1} a_j)/m] \leq [ta_i/m].$$

This implies that, whenever A is super-increasing, then so is every $A(k)$.

Lemma 2 *Assume that* $A \xrightarrow[(t,m)]{} B$ *holds for some* B. *Then* $A(k) \xrightarrow[(t,m+kt)]{} B$ *holds for all* k. *If* B *is* (A,t,m)*-reachable (resp.* (A,t,m)*-super-reachable), then, for all* k, B *is also* $(A(k),t,m+kt)$*-reachable (resp.* $(A(k),t,m+kt)$*-super-reachable).*

Proof. Denoting $B = (b_1, \ldots, b_n)$, we infer by the assumption:

$$b_i = (ta_i, \operatorname{mod} m), \text{ for } 1 \le i \le n.$$

Clearly, $(t, m+kt) = 1$. By (1), for all k,

$$t(a_i + k \cdot \lfloor ta_i/m \rfloor) = b_i + \lfloor ta_i/m \rfloor \cdot m + \lfloor ta_i/m \rfloor \cdot kt = b_i + \lfloor ta_i/m \rfloor (m+kt).$$

Since by the definition of b_i we have $b_i < m + kt$, we conclude that

$$(t(a_i + k \cdot \lfloor ta_i/m \rfloor), \operatorname{mod}(m+kt)) = b_i.$$

Therefore,

$$A(k) \xrightarrow[(t,m+kt)]{} B. \tag{6}$$

By Lemma 1, $A(k)$ is increasing if A is and hence, by (6), the claim concerning reachability follows.

Assume that B is (A,t,m)-super-reachable. By Lemma 1, each $A(k)$ is super-increasing. Moreover,

$$\sum_{i=1}^{n} a_i < m.$$

This implies that

$$\sum_{i=1}^{n}(a_i + k\lfloor ta_i/m \rfloor) < m + \sum_{i=1}^{n} k \cdot \lfloor ta_i/m \rfloor \le m + k\lfloor t(a_1 + \ldots + a_n)/m \rfloor$$

$$\le m + k \cdot \lfloor t \rfloor = m + kt.$$

Consequently, B is $(A(k), t, m+kt)$-super-reachable. □

It is an immediate consequence of Lemma 2 that every super-reachable vector can be obtained from infinitely many super-increasing vectors by modular multiplication with a big enough modulus. The special case, where $\lfloor ta_i/m \rfloor = 0$ for all i, can be easily handled separately.

We now investigate the question of which triples (A,t,m) possess goals. Recall that every violation point i of A satisfies, by definition,

$$a_i \le \sum_{j=1}^{i-1} a_j. \tag{7}$$

Assume that also

$$\lfloor ta_1/m \rfloor + \ldots + \lfloor ta_{i-1}/m \rfloor < \lfloor ta_i/m \rfloor. \tag{8}$$

(Observe that (7) and (8) are by no means contradictory.) Then the smallest integer x such that

$$\sum_{j=1}^{i-1} a_j + x \sum_{j=1}^{i-1} \lfloor ta_j/m \rfloor < a_i + x \cdot \lfloor ta_i/m \rfloor \tag{9}$$

is called the *rescuer* of i. Explicitly,

$$x = \lfloor((\sum_{j=1}^{i-1} a_j) - a_i)/(\lceil ta_i/m\rceil - \sum_{j=1}^{i-1}\lceil ta_j/m\rceil)\rfloor + 1.$$

By (7) and (8), x is a positive integer.

If (8) holds for every violation point i, then the *rescuer* of A is defined to be the maximum of the rescuers of all violation points i.

We consider, next, the situation where the modulus is not big enough:

$$m \le \sum_{i=1}^{n} a_i. \tag{10}$$

Assume that also

$$\sum_{i=1}^{n}\lceil ta_i/m\rceil < t. \tag{11}$$

Then the smallest integer y such that

$$\sum_{i=1}^{n} a_i + y\sum_{i=1}^{n}\lceil ta_i/m\rceil < m + yt \tag{12}$$

is called the *rescuer* of m. Explicitly,

$$y = \lfloor((\sum_{i=1}^{n} a_i) - m)/(t - \sum_{i=1}^{n}\lceil ta_i/m\rceil)\rfloor + 1.$$

We infer by (10) and (11) that the rescuer of m is a positive integer. It is important to notice that if (9) (resp. (12)) holds for some x (resp. y) then it holds for all integers $> x$ (resp. $> y$) as well. This means that if i' is rescued by k', that is, i' is not a violation point in $A(k')$, then i' is not a violation point in any $A(k), k > k'$. Hence, if we have to rescue several numbers (possibly including m), then we may go further in the growing sequence until all of them have been rescued (if ever). For the sake of completeness, we say that 0 is the rescuer of i (resp. m) if (7) resp. (10)) does not hold.

Lemma 3 *A triple (A, t, m) possesses a goal iff (8) holds whenever (7) holds and, moreover, (11) holds in case (10) holds. If these conditions are satisfied, the goal is $(A(k_0), t, m + k_0t)$, where k_0 is the maximum of the rescuers of A and m.*

Proof. If k_0 is defined as in the statement of the lemma, then $A(k_0)$ is superincreasing (because it has no violation points) and $m + k_0t$ is greater than the sum of the components of $A(k_0)$. The definition of k_0 guarantees that we obtain the smallest number satisfying these conditions. On the other hand, if some i satisfies (7) but in (8) we have $\ge$ instead of $<$, then i is a violation point in every $A(k)$. Similarly, if (10) holds but (11) does not hold, then for all k,

$$\sum_{i=1}^{n}(a_i + k\lceil ta_i/m\rceil) \ge m + kt.$$

Hence, the modulus is too small in every triple of the growing sequence. □

We now give some illustrations. The examples are given in terms of tables, where A, t, m, B and the goal are listed. Here B is the result of modular multiplication (that is, the vector satisfying (2)). By the second sentence of Lemma 3, the goal gives items showing that B is super-reachable. If no goal exists, we use the abbreviations NR($i = i'$) and NR(m) to mean that a violation point $i = i'$ or too small a modulus m does not possess a rescuer (that is, (8) or (11) is not satisfied). In some examples there may be several such failures. The existence of one failure already shows that there is no goal.

Example 1 We begin with some vectors considered above in Section 2.

A	t	m	B	Goal
(1,2,3)	4	5	(4,3,2)	$k = 2$, (1,4,7),4,13
(1,4,7)	3	8	(3,4,5)	NR (m):$0 + 1 + 2 \geq 3$
(1,5,6)	7	8	(7,3,2)	$k = 1$ rescuer of $i = 3$, NR (m)
(1,3,5)	4	9	(4,3,2)	$k = 1$, (1,4,7),4,13

We continue with some other illustrations. Different cases concerning which numbers can be rescued will be included.

A	t	m	B	Goal
(1,3,6)	3	7	(3,2,4)	NR (m)
(2,3,4)	5	6	(4,3,2)	NR ($i = 3$), NR (m)
(1,2,3)	5	6	(5,4,3)	$k = 1$, (1,3,5),5,11
(1,5,12)	8	13	(8,1,5)	NR (m)
(1,2,10)	8	15	(8,1,5)	Own goal
(1,8,13,36,57)	87	200	(87,96,131,132,159)	$k = 2$,(1,14,23,66,105),87,374
(1,34,67)	97	100	(97,98,99)	$k = 3$,(1,130,259,),97,391
(1,15,29,44)	93	100	(93,95,97,92)	$k = 2$,(1,41,81,124),93,286
(2,3,5,8)	4	9	(8,3,2,5)	NR ($i = 4$), NR (m), $k = 1$ rescuer of $i = 3$.

4 Fundamental lemmas II

The first lemma in this section deals with an interplay between the multiplier and the modulus. We then discuss properties of diminishing sequences. Finally, growing and diminishing sequences are tied together.

Lemma 4 *Assume that max $B < t < m$ and*

$$A \underset{(t,m)}{\longrightarrow} B \,(\text{resp. } A \underset{(M,t,m)}{\longrightarrow} B) \tag{13}$$

holds. Then the items $A', t' \leq$ max B and m' defined below satisfy

$$A' \underset{(t',m')}{\longrightarrow} B' \,(\text{resp. } A' \underset{(M',t',m')}{\longrightarrow} B'). \tag{14}$$

If B is super-reachable, then B is (A', t', m')- super-reachable with $t' \leq$ max B.

Proof. Assume that in (13) max $B < t < m$. We define another triple (A_1, t_1, m_1) such that $t_1 < t$ and (14) holds with (A', t', m') replaced by (A_1, t_1, m_1). (14) with $t' \leq$ max B is then established by repeating this construction as many times as necessary.

Our definition interchanges the multiplier and the modulus as follows:

$$m_1 = t, t_1 = (-m, \bmod t), A_1 = ([ta_1/m], \ldots, [ta_n/m]).$$

Clearly, $t_1 < t$ and $(t_1, m_1) = 1$. By (1) and our assumption (13) we obtain, for $1 \leq i \leq n$,

$$t_1[ta_i/m] \equiv b_i - ta_i \equiv b_i \pmod t.$$

Since $b_i \leq$ max $B < t$, we may write further

$$(t_1[ta_i/m], \bmod t) = b_i.$$

Corresponding to the "resp."-statement in parentheses in (13) and (14), we still have to show that if m exceeds the sum of the components of A, then t exceeds the sum of the components of A_1. But this is clear. If $m > \sum_{i=1}^{n} a_i$, then also

$$t > \sum_{i=1}^{n} ta_i/m \geq \sum_{i=1}^{n} [ta_i/m].$$

To prove the last sentence of Lemma 4, we show that if A is super-increasing then so is A_1. If A is super-increasing, we have for $2 \leq i \leq n$,

$$\sum_{j=1}^{i-1} ta_j/m > ta_i/m.$$

(The original inequality is multiplied by t/m.) Hence,

$$\sum_{j=1}^{i-1} [ta_j/m] \leq [ta_i/m]. \tag{15}$$

Assume that we have equality in (15). Then

$$\sum_{j=1}^{i-1} m[ta_j/m] = m[ta_i/m].$$

Applying again (1) we obtain

$$\sum_{j=1}^{i-1} (ta_j - b_j) = ta_i - b_i$$

and, hence,

$$b_i - \sum_{j=1}^{i-1} b_j = t(a_i - \sum_{j=1}^{i-1} a_j).$$

The coefficient of t on the right side is positive and, consequently,

$$t \leq b_i - \sum_{j=1}^{i-1} b_j < b_i \leq \max B,$$

which contradicts the assumption $t > \max B$. This implies that we must have strict inequality in (15). Hence, A_1 is super-increasing. □

As an illustration of the technique of Lemma 4, observe first that the vector (7,3,2) is ((7,15,38), 73, 84)-super- reachable. Here the multiplier 73 is much too big. The technique yields, successively, the following triples.

$$((6,13,33),62,73),((5,11,28),51,62),$$

$$((4,9,23),40,51),((3,7,18),29,40),$$

$$((2,5,13),18,29),((1,3,8),7,18).$$

The vector (7,3,2) is super-reachable with respect to all of these triples. In the last triple the multiplier is sufficiently small.

Similarly, the vector (46,45,40,30) is ((4,5,10,20), 49,50)-super-reachable. It is also super-reachable for each of the triples

$$((3,4,9,19),48,49),((2,3,8,18),47,48),((1,2,7,17),46,47).$$

In case of the ((2,5,8,17),32,33)-super-reachability of the vector (31,28,25,16) only one interchange between multiplier and modulus makes the new multiplier sufficiently small. The vector (31,28,25,16) is also ((1,4,7,16),31,32)- super-reachable.

In the following lemma we use the same notation as in the definition of diminishing sequences.

Lemma 5 *Every triple* $(A(-k), t, m-kt), 0 \leq k \leq s$, *in the diminishing sequence associated with the triple* (A, t, m) *satisfies*

$$A(-k) \xrightarrow[(t,m-kt)]{} B. \tag{16}$$

Moreover, if A *is increasing, then so is every vector* $A(-k), 0 \leq k \leq s$.

Proof. We prove the first sentence by induction on k. For $k = 0$, (16) holds by the definition of the diminishing sequence. Assume that (16) holds and we still have

$$m - kt > 2 \max B. \tag{17}$$

We will show that

$$A(-k-1) \xrightarrow[(t,m-(k+1)t)]{} B. \tag{18}$$

Denote $A(-k) = (d_1, \ldots, d_n)$. Then the ith component of $A(-k-1), 1 \leq i \leq n$, is

$$d_i - [td_i/(m-kt)].$$

Multiplying this by t and using (1) and (16), we obtain

$$td_i - t[td_i/(m-kt)] = b_i + (m-kt)[td_i/(m-kt)] - t[td_i/(m-kt)] =$$

$$= b_i + (m-(k+1)t)[td_i/(m-kt)] \equiv b_i (\bmod m - (k+1)t).$$

By (17) and the assumption $t \leq \max B$ made in the definition of the diminishing sequence,

$$m - (k+1)t > \max B \geq b_i.$$

This implies that

$$(t(d_i - [td_i/(m-kt)]), \bmod m - (k+1)t) = b_i$$

and, consequently, (18) holds.

The second sentence of Lemma 5 is established also by induction on k. Assume that $A = A(-0)$ is increasing. We make the inductive hypothesis that $A(-k) = (d_1, \ldots, d_n)$ is increasing and (17) holds. Denote $A(-k-1) = (e_1, \ldots, e_n)$. Because of (17) and the inequality $t \leq \max B$, we obtain

$$m - kt > 2t. \tag{19}$$

Consider now an arbitrary i, $1 \leq i \leq n-1$. Since $A(-k)$ is increasing,

$$d_{i+1} = d_i + \alpha \text{ for some } \alpha \geq 1.$$

Assume first that $\alpha > 1$. Then

$$e_{i+1} = d_i + \alpha - [t(d_i+\alpha)/m - kt)] \geq d_i + \alpha - (1 + [td_i/(m-kt)] + [t\alpha/(m-kt)])$$
$$= e_i + (\alpha - 1) - [t\alpha/(m-kt)] > e_i.$$

Here the first inequality follows because always $[x+y] \leq [x] + [y] + 1$, and the second because, by (19),

$$[t\alpha/(m-kt)] \leq t\alpha/(m-kt) < \alpha/2.$$

Assume, secondly, that $\alpha = 1$. In this case $[t\alpha/(m-kt)] = 0$. If

$$[t(d_i+1)/(m-kt)] = [td_i/(m-kt)],$$

we obtain $e_{i+1} > e_i$. Hence, suppose that

$$[t(d_i+1)/(m-kt)] = [td_i/(m-kt)] + 1. \tag{20}$$

(By the above estimate for e_{i+1} there are no other possibilities. (20) would imply that $e_{i+1} = e_i$.) Denote the right side of (20) by $\beta + 1$. Hence,

$$(m-kt)\beta \leq td_i < (m-kt)(\beta+1) \leq t(d_i+1).$$

Assume that $td_i < (m-kt)(\beta + \frac{1}{2})$. Hence, by (19),

$$td_i + t < (m-kt)(\beta + \frac{1}{2}) + t = (m-kt)(\beta+1) + t - \frac{m-kt}{2} < (m-kt)(\beta+1),$$

a contradiction. Hence, $td_i \geq (m-kt)(\beta + \frac{1}{2})$. But now, by (16),

$$b_i = td_i - \beta(m-kt) \geq (m-kt)/2.$$

This implies that $m - kt \leq 2b_i \leq 2\max B$, contradicting (17). This shows that (20) cannot hold. $\square$

It is important to note that certain properties preserved by the growing sequences are not preserved by the diminishing sequences. A may be super-increasing although the other vectors in the diminishing sequence are not. For instance, choose

$$A = (1, 14, 23, 66, 105), t = 87, m = 374,$$

implying that $B = (87, 96, 131, 132, 159)$ and, hence $t \leq \max B$ and $m > 2 \max B$. Now

$$A(-1) = (1, 11, 18, 51, 81),$$

which is not super-increasing. Similarly, we see that

$$(1,4,7) \underset{(M,4,13)}{\longrightarrow} (4,3,2)$$

but when we go to the first triple in the diminishing sequence, we observe that not

$$(1,3,5) \underset{(M,4,9)}{\longrightarrow} (4,3,2)$$

because $9 = 1 + 3 + 5$. Thus, the M-relation is not preserved. In the second triple ((1,2,3),4,5,) of the same diminishing sequence neither is the M-relation satisfied nor is the vector super-increasing. Such negative results are natural in view of the following lemma and reflect the fact that some properties are rescued from a certain point on in the growing sequence. The same properties are lost at this point in the diminishing sequence.

In the statement of the following lemma, the notation A, t, m, s, B is the same as in the definition of the diminishing sequence.

Lemma 6 *Assume that $(A(-k), t, m - kt), 0 \leq k \leq s$, is the diminishing sequence associated with the triple (A, t, m). Denote $A(-s) = C$. Consider the initial segment*

$$(C(k), t, m - st + kt), 0 \leq k \leq s,$$

of the growing sequence associated with the triple $(C = C(0), t, m - st)$. Then, for each k such that $0 \leq k \leq s$,

$$C(k) = A(-(s - k)). \tag{21}$$

Proof. Our intention is to use induction on k. For this purpose, it is useful to denote

$$A(-(s - k)) = (a_1^k, \ldots, a_n^k), C(k) = (c_1^k, \ldots, c_n^k),$$

for $0 \leq k \leq s$. Clearly, $A(-(s - s)) = (a_1, \ldots, a_n) = A$. We consider an arbitrary k and i in their respective ranges. To simplify notation, we write $a^k = a_i^k$ and $c^k = c_i^k$. By the definitions of growing and diminishing sequences,

$$c^{k+1} = c^k + \lfloor tc^0/\alpha \rfloor \text{ and } a^{k+1} = a^k + \lfloor ta^{k+1}/(\alpha + (k+1)t) \rfloor, \tag{22}$$

where $\alpha = m - st$. We have to establish $c^k = a^k$, for all k with $0 \leq k \leq s$, in order to establish (21). By the choice of $C(0)$, we have $a^0 = c^0$. Using (22), we show that $c^1 = a^1$. Thus, we have to prove that

$$\lfloor ta/\alpha \rfloor = \lfloor ta^1/(\alpha + t) \rfloor, \tag{23}$$

where we denote $c^0 = a^0 = a$. By Lemmas 2 and 5,

$$ta^1 \equiv tc^1 \text{ and, hence, } a^1 \equiv c^1 \pmod{\alpha + t}.$$

Because $a^0 = c^0$, we infer that

$$[ta/\alpha] \equiv [ta^1/(\alpha+t)] \,(\operatorname{mod} \alpha + t). \tag{24}$$

(24) can hold without (23) holding only in case that the absolute value of the difference between the two bracket expressions is a positive multiple of $\alpha + t$. We prove that this is impossible by showing that both of the bracket expressions (which clearly are nonnegative) are less than $\alpha + t$. Since $\alpha > \max C(0) = \max A(-s) \geq a$, we obtain

$$[ta/\alpha] < t < \alpha + t.$$

The bracket expression on the right side of (24) is estimated by repeated use of the principle $[x] \leq x$, yielding when we denote $t/(\alpha+t) = x$

$$\begin{aligned}[ta^1/(\alpha+t)] \leq ta^1/(\alpha+t) &= \frac{t}{\alpha+t}(a + [ta^1/(\alpha+t)]) \\ &\leq \frac{t}{\alpha+t}(a + \frac{t}{\alpha+t}(a + [ta^1/(\alpha+t)]))\ldots \\ &\leq a(x + x^2 + \ldots + x^p) + x^{p+1}a^1 \\ &\leq a/(1-x) + x^{p+1}a^1 = a + at/\alpha + x^{p+1}a^1 \\ &< \alpha + t + x^{p+1}a^1.\end{aligned}$$

This holds for arbitrarily large p, which means that the term $x^{p+1}a^1$ can be made arbitrarily small. Consequently,

$$[ta^1/(\alpha+t)] < \alpha + t.$$

By (24), (23) holds. We have shown that $a^1 = c^1$.

The inductive step from $a^k = c^k$ to $a^{k+1} = c^{k+1}$ is now very easy. We consider only the initial segment of the diminishing sequence to the triple $(A(-(s-k)), t, m-(s-k)t)$. We start the growing sequence from this triple. Also now we have to establish (23), where now $\alpha = m - (s-k)t, a = a^k = c^k$ and $a^1 = a^{k+1}$. The proof is exactly the same as above. This completes the induction and, hence, (21) holds.

5 Super-reachability

We are now in the position to establish one of our main results.

Theorem 3 *A knapsack vector B is super-reachable iff B is (A, t, m)-reachable, where $t \leq \max B, m \leq 2\max B$ and the triple (A, t, m) possesses a goal.*

Proof. We already have developed all the necessary technical apparatus. The "if"-part follows by Lemma 2 and the definition of the goal. Lemma 3 gives a simple method for deciding whether or not a given triple possesses a goal.

For the "only if"-part, assume that B is super-reachable. By Lemma 4, B is (A, t, m)-super-reachable with $t \leq \max B$. If $m \leq 2\max B$, we are finished. Otherwise, we form the diminishing sequence

$$(A(-k), t, m - kt), 0 \leq k \leq s,$$

where $m - st \leq 2 \max B$. Since A is increasing, we conclude, by Lemma 5, that B is $(A(-s), t, m - st)$-reachable and, by Lemma 6, that the triple $(A(-s), t, m - st)$ possesses a goal. □

The algorithm due to Theorem 3 can be described as follows. Given B, choose $\max B < m \leq 2 \max B$ and $u < m$ with $(u, m) = 1$. Check whether the vector A satisfying

$$B \xrightarrow[(u,m)]{} A$$

is increasing and $u^{-1} = t \leq \max B$. If not, choose another pair (u, m). Else check whether the triple (A, t, m) possesses a goal. If not, choose another pair (u, m). Otherwise, B is super-increasing. The goal also gives a super-increasing vector, multiplier and modulus showing this.

The time complexity of the algorithm is estimated in [4] . Complexity in terms of max B is at most cubic. Complexity in terms of n depends on the upper bound for max B in terms of n. Such upper bounds are given, for instance, in [1] and [5]. They are always arbitrary and leave out most of the instances, whereas the algorithm of Theorem 3 works independently of any bounds, for max B. Reductions in the estimates can possibly be made by a more detailed analysis of the number of successful pairs (u, m).

Example 2 We now give some illustrations of the algorithm of Theorem 3. Again, for the sake of readability, the illustrations are very small in size. We consider first the vectors (1,10,8) and (4,1,6) shown 2- and 3-hyper-reachable in Section 2. Consider first the vector (4,1,6). The pairs (u, m) to be investigated are listed in the following table.

m	12	11	10	9	8	7
u	5,7,11	2,3,...,10	3,7,9	2,4,5,7,8	3,5,7	2,3,4,5,6

The next table shows the actual application of the algorithm. The leftmost column lists all the pairs (u, m) which might lead to success, that is, $u^{-1} = t \leq \max B = 6$ (inverse is taken modulo m), and the vector A obtained from $B = (4, 1, 6)$ by modular multiplication due to (u, m) is increasing. The items t and A are listed in the next two columns. If A is not super- increasing or $m \leq a_1 + a_2 + a_3$, we investigate whether or not the violation point i' (here only $i' = 3$ is possible) and the modulus m can be rescued. If they can, the last column indicates the value of k for which the goal is reached in the growing sequence associated with the triple (A, t, m). The last column also indicates the three items of the goal. If at least one of the numbers cannot be rescued, we use the abbreviations NR($i = i'$) and NR(m) as before.

u, m	$t = u^{-1}$	A	Goal
3,11	4	(1,3,7)	$k = 1$, (1,4,9),4,15
9,11	5	(3,9,10)	NR($i = 3$), NR(m)
5,8	5	(4,5,6)	NR($i = 3$), NR(m)
2,7	4	(1,2,5)	$k = 2$, (1,4,9),4,15

It is interesting to note that in both cases leading to success we obtain the same triple ((1,4,9),4,15). Thus, this triple can be visualized as the minimal or prime triple for which (4,1,6) is super-reachable. More specifically, whenever (4,1,6) is (A, t, m)-super-reachable, then $t \geq 4$ and $m \geq 15$. This follows because the algorithm would produce any smaller values of t and m. Of course, m can be made arbitrarily large in the growing sequence. Also t can be made larger by applying an argument similar to that used in Lemma 4 in the reverse order.

The vector $(4,1,6) = B$ shows also that it is in general not sufficient to investigate candidates $m \leq 2\max B$, without taking into account the growing sequence. If this would be done for (4,1,6), we would never find the solution. However, it is possible to obtain the following general result.

Theorem 4 *A knapsack vector B is super-reachable iff B is (A,t,m')-super-reachable where $t \leq \max B$ and $m' \leq 2\max B(1+\max B)$.*

Proof. The upper bound for t is obtained by Lemma 4 exactly as before. To obtain an upper bound for the modulus, we have to deduce an upper bound for the moduli $m+kt$ in the initial segment of the growing sequence consisting of triples up to the goal. We know that $t \leq \max B$ and $m \leq 2\max B$. Since a goal is reached, the difference between the sum of the components of the vector and the modulus decreases at least by one in every step from a triple to the next triple in the growing sequence. This holds true also as regards the difference defined by any violation point. The goal is reached when all of these differences are negative. Hence, the goal is reached latest in m (= the original modulus) steps, implying that $k \leq 2\max B$. Consequently, $m' \leq 2\max B + (2\max B)\max B$. □

The statement of Theorem 4 is simpler than that of Theorem 3. However, the resulting algorithm is considerably less efficient, as shown even by examples of small size.

Example 3 Consider now the vector (1,10,8), shown 2-hyper-reachable in Section 2. We have to consider moduli $m \leq 20$. For each m, we must have $u < m$ and $(u,m) = 1$. The following table of pairs (u,m) that may lead to success is obtained exactly as in Example 2.

u,m	$t = u^{-1}$	A	Goal
7,20	3	(7,10,16)	NR $(i=3)$, NR (m)
9,20	9	(9,10,12)	NR $(i=3)$, NR (m)
2,17	9	(2,3,16)	NR (m)
6,17	3	(6,9,14)	NR $(i=3)$, NR (m)
5,14	3	(5,8,12)	NR $(i=3)$, NR (m)
3,13	9	(3,4,11)	NR (m)
4,11	3	(4,7,10)	NR $(i=3)$, NR (m)
5,11	9	(5,6,7)	NR $(i=3)$, NR (m)

We conclude that (1,10,8) is not super-reachable. Hence, we have established the following result.

Theorem 5 *There are 2-hyper-reachable knapsack vectors that are not super-reachable.*

It is an open problem whether or not r-hyper-reachable vectors form a strictly increasing hierarchy with r increasing. Other examples of strictly 2-hyper-reachable vectors are easy to construct.

Example 4 We now give the table for each permutation of the vector (2,3,4). In each case only values $m \leq 8$ have to be considered.

(2,3,4): No candidates (u,m)

	u, m	$t = u^{-1}$	A	Goal
(3,4,2):	3,8	3	(1,4,6)	NR (m)
	2,5	3	(1,3,4)	$k = 1$ rescues $i = 3$, NR (m)
(4,2,3):	2,7	4	(1,4,6)	NR (m)
(2,4,3):	4,7	2	(1,2,5)	$k = 2$, (1,2,7),2,11
	3,5	2	(1,2,4)	$k = 3$, (1,2,7), 2,11
(3,2,4):	(5,7)	3	(1,3,6)	NR (m)
(4,3,2)	(4,5)	4	(1,2,3)	$k = 2$, (1,4,7),4,13

The study of (4,3,2) is interesting because it shows that we cannot reject non-injective candidates A in spite of Theorem 1. This is due to the fact that injectivity can be gained later on in the growing sequence.

We now investigate similarly all permutations of the vector (1,2,4).

(1,2,4): super-increasing

	u, m	$t = u^{-1}$	A	Goal
(1,4,2):	3,8	3	(3,4,6)	NR ($i = 3$), NR (m)
	2,5	3	(2,3,4)	NR ($i = 3$), NR (m)
(2,1,4):	5,7	3	(3,5,6)	NR ($i = 3$), NR (m)
(2,4,1):	4,7	2	(1,2,4)	$k = 1$, (1,2,5),2,9
	3,5	2	(1,2,3)	$k = 2$, (1,2,5),2,9
(4,1,2):	2,7	4	(1,2,4)	$k = 1$, (1,3,6),4,11
(4,2,1):	4,5	4	(1,3,4)	$k = 1$ rescues $i = 3$, NR (m)

Summarizing we obtain the following result.

Theorem 6 *Consider knapsack vectors with all components ≤ 4. Exactly the following ones are super-reachable:*

$$(2,4,3),(4,3,2),(1,2,4),(2,4,1),(4,1,2)$$

Proof. By Theorem 1, no permutation of any of the vectors (1,3,4), (1,2,3), (1,2,3,4) can be super-reachable. The remaining cases were classified in Example 4. □

6 Consequences and modifications

Several other decidability results can be obtained using our basic technique of growing and diminishing sequences. We mention a minimization result concerning the multiplier and the modulus.

Theorem 7 *Assume that B is super-reachable. Then the smallest m (resp. the smallest t) such that B is (A, t, m)-super-reachable for some A and t (resp. A and m) is effectively computable.*

Proof. By Theorem 3 or Theorem 4, some triple (A, t, m) is obtained. A straightforward way of minimizing the modulus would be a systematic search through all values $m' < m$. For each m', it suffices to test the finitely many triples (A', t', m'), where $t' < m'$ and the sum of the components of the super-increasing A' is less than m'. However, a much more efficient algorithm (running in time at most cubic in terms of max B) is obtained by Theorem 3: the smallest modulus can be found from the triples produced by Theorem 3. The same holds true as regards the smallest multiplier t. We have presented several examples, where it is necessary to go into the growing sequence in order to find the smallest modulus, as well as examples, where the smallest multiplier is considerably less than max B. □

A consequence of Theorem 3, apparent also in the examples above, is that a vector B is not super-reachable if there are no candidates (A, t, m), where A is increasing, $t = u^{-1} \leq \max B, m \leq 2 \max B$ and A results from B by modular multiplication using u and m. The special case, where B itself is increasing but not super-increasing, is interesting. Considering small examples, one is tempted to conjecture that growing sequences do not at all come into use, that is, one may restrict the attention to vectors A reachable from B by modular multiplication using u and m. However, the following example shows that this conjecture is false. Choose $B = (87, 96, 131, 132, 159), m = 200$ and $n = 23$. Then $t = 87$ and $A = (1, 8, 13, 36, 57)$, which is not super-increasing. However, the triple (A, t, m) possesses the goal

$$((1, 14, 23, 66, 105), 87, 374),$$

reached for $k = 2$. Here $374 > 2$ max B.

The following result can be obtained along these lines.

Theorem 8 *If* $B = (b_1, b_2, b_3)$ *is increasing and super-reachable, then* B *is* (A, t, m)*-super-reachable, for some* $t \leq \max B, m \leq 2 \max B$.

Proof. There must be an increasing A such that

$$B \underset{(u,m)}{\longrightarrow} A$$

where $m \leq 2 \max B, t = u^{-1} \leq \max B$. Suppose that no such A is super-increasing. Consider an arbitrary $A = (a_1, a_2, a_3)$. Hence, 3 is a violation point: $a_3 \leq a_1 + a_2$. This implies that $ta_3 \leq ta_1 + ta_2$. Denote $ta_i = b_i + \alpha_i m, i = 1, 2, 3$. (Hence, $[ta_i/m] = \alpha_i$.) We obtain

$$b_3 + \alpha_3 m \leq b_1 + b_2 + (\alpha_1 + \alpha_2)m.$$

If now $\alpha_3 > \alpha_1 + \alpha_2$, we obtain further

$$b_3 + m \leq b_3 + (\alpha_3 - \alpha_1 - \alpha_2)m \leq b_1 + b_2 + (\alpha_1 + \alpha_2 - \alpha_3)m + (\alpha_3 - \alpha_1 - \alpha_2)m =$$
$$= b_1 + b_2 < b_1 + m.$$

Consequently, $b_3 < b_1$, which contradicts the assumption of B being increasing. This implies that $\alpha_3 \leq \alpha_1 + \alpha_2$, which shows that the triple (A, t, m) possesses no goal. Since A was arbitrary, we conclude that B is not super-reachable, contrary to the assumption. □

It was seen above that the result of Theorem 8 does not hold true if the number n of the components of the vectors equals 5. It is an open problem whether or not the result holds for $n = 4$.

Our final example in this section is of a somewhat different nature.

Example 5 Shamir's algorithm (see [5] or [2]) is based on the assumption that the given vector B is super-reachable. The algorithm usually produces an interval such that, whenever the number u/m written in lowest terms lies in this interval, then B is (A, t, m)-super-reachable, $t = u^{-1}$ and

$$B \underset{(u,m)}{\longrightarrow} A.$$

Without explaining any details of Shamir's algorithm, we show by a couple of examples how one can go back to the algorithm of Theorem 3.

Consider the vector $B = (7, 3, 2)$. We get an open interval $(5/7, 3/4)$. The number $8/11$ in this interval yields $A = (1, 2, 5)$. This sows that B is $(A, 7, 11)$-super-reachable. Here both 7 and 11 are within the limits of Theorem 3 and, thus, the result is obtained by the "first part" of our algorithm, where one does not use growing sequences. The number $41/56$ in this interval yields $A = (7, 11, 26)$. Since 41 is its own inverse, B is ((7,11,26),41,56)-super-reachable. The multiplier 41 is, however, too big to be reached by the algorithm of Theorem 3. Using Lemma 4, we get successively the following triples:

$$((5, 8, 19), 26, 41), ((3, 5, 12), 11, 26), ((1, 2, 5), 7, 11).$$

Here the last triple (in fact, the same as the one obtained for $8/11$) falls within the size limits of Theorem 3.

As regards the number $61/84$ from the interval in question, the procedure is slightly different. The inverse of 61 is 73 and, hence, B is ((7,15,38),73,84)-super-reachable. Again, the multiplier is too big. Lemma 4 yields, in succession, the triples

$$((6, 13, 33), 62, 73), ((5, 11, 28), 51, 62),$$

$$((4, 9, 23), 40, 51), ((3, 7, 18), 29, 40),$$

$$((2, 5, 13), 18, 29), ((1, 3, 8), 7, 18).$$

In the last triple the multiplier $t = 7$ satisfies $t \leq \max B$. In fact, we already carried out the computation this far after Lemma 4. However, $m > 2 \max B$ and cannot be obtained in the first part of the algorithm. Taking one step in the diminishing sequence we obtain our old friend ((1,2,5),7,11), which completes our argument.

In the example considered in [2],

$$B = (43, 129, 215, 473, 903, 302, 561, 1165, 697, 1523)$$

is the vector to be analyzed. Shamir's algorithm produces the interval (1/43, 36/1547). Choosing $u = 37$ and $m = 1590$, we get the number $37/1590$ in this interval, as well as the vector

$$A = (1, 3, 5, 11, 21, 44, 87, 175, 349, 701)$$

which is super-increasing. Now $u^{-1} = t = 43$ and, thus, both t and u lie within the bounds of the first part of the algorithm of Theorem 3. In fact, the solution obtained equals the one used by the cryptosystem designer in [2] .

Consider next $u = 72$ and $m = 3095$. We get the vector

$$A = (1, 3, 5, 11, 21, 79, 157, 315, 664, 1331).$$

Also now $t = 43$ but $m > 2 \max B$. When we go two steps back in the diminishing sequence, we get the triple

$$((1, 3, 5, 11, 21, 77, 153, 307, 646, 1295), 43, 3009).$$

Now also m is within the size limits.

7 Permutations

For a cryptanalyst it is certainly sufficient to find a permutation of a publicized vector B that is super-reachable. When such a permutation is known, cryptanalysis works as before - only the inverse permutation has to be applied to the plaintext bit vectors.

Let us call a vector B *permutation-super-reachable* iff some permutation of B is super-reachable. For instance, it was seen in Example 3 that (1,10,8) is not super-reachable. Clearly, it is permutation-super-reachable. By our theory it is easy to see that every injective (a_1, a_2, a_3) is permutation-super-reachable. The following result is established exactly as Theorem 1.

Theorem 9 *Every permutation-super-reachable vector is injective.*

Permutations were investigated already in Example 4. The following example is of a similar nature.

Example 6 We use the same notation as in Example 4 to classify the permutations of (3,4,5).

	u, m	$t = u^{-1}$	A	Goal
(3,4,5):	3,8	3	(1,4,7)	NR (m)
(3,5,4):	7,10	3	(1,5,8)	NR (m)
	5,7	3	(1,4,6)	NR (m)
(4,3,5):	5,9	2	(2,6,7)	NR ($i = 3$), NR (m)
	7,9	4	(1,3,8)	NR (m)
	4,7	2	(2,5,6)	NR ($i = 3$), NR (m)
(4,5,3):	2.7	4	(1,3,6)	NR (m)
(5,3,4):	2,9	5	(1,6,8)	NR (m)
	3,7	5	(1,2,5)	$k = 2$, (1,4,11),5,17
(5,4,3):	5,8	5	(1,4,7)	NR (m)
	5,6	5	(1,2,3)	$k = 1$, (1,3,5),5, 11

Thus, only (5,3,4) and (5,4,3) are super-reachable.

8 Hyper-reachability

Various decidability results and polynomial-time algorithms concerning hyper-reachability can be obtained using the techniques developed above. We mention here some such results. All of them concern r-hyper-reachability for a fixed or bounded r. This is basically due to the fact that a characterization of hyper-reachability is missing. Do the r-hyper-reachable sets of vectors form an infinite hierarchy (when r is growing)? It is conceivable that, for some target vectors B, the "derivation chain" is arbitrarily long with irregular fluctuations in the sizes of the intermediate vectors and moduli.

The following Theorems 10-12 correspond to Theorems 3,4 and 7, respectively.

Theorem 10 *It is decidable of a given knapsack vector B and positive integer r whether or not B is r-hyper-reachable.*

Proof. Consider first the case $r = 2$. Then B is 2-hyper-reachable iff there exist t, m, t', m', C and a super-increasing A such that

$$A \underset{(M,t',m')}{\longrightarrow} C \underset{(M,t,m)}{\longrightarrow} B. \tag{25}$$

The method of Theorem 3 is now applicable with the exception that we cannot use Lemma 4 in connection with C. The construction leading from (13) to (14) is valid but does not necessarily preserve the super-reachability of the vectors involved. In (25) C is super-reachable, whereas the vector obtained form C by the construction of Lemma 4 might not be super-reachable. However, the strict separation $t \leq \max B, m > 2\max B$ is needed only in (19) to prove that diminishing sequences preserve the property of being increasing. We do not need this property in connection with C. The proofs of the Lemmas 2,3 (where the goal is defined only for m), 5 (where the requirement $t \leq \max B$ is omitted from the definition of a diminishing sequence) and 6 remain valid.

We proceed as follows. For arbitrary $u < m \leq 2\max B$, we form the vector $E = (e_1, \ldots, e_n)$ such that

$$B \underset{(u,m)}{\longrightarrow} E.$$

(E need not be increasing and not necessarily $u^{-1} = t \leq \max B$.) If $m > \sum_{i=1}^{n} e_i$, the vector E qualifies as a candidate for C. Otherwise, we test by Lemma 3 whether or not the modulus can be rescued in the growing sequence associated with (E, t, m). If it can, then the resulting vector E' qualifies as a candidate for C. Then all candidates are obtained according to Theorem 3. If originally in (25) $m > 2\max B$, a modulus of the right size is obtained in the diminishing sequence. The result need not be super-reachable because the original C is recovered in the growing sequence.

The case of a general r is now obvious by induction. Assuming the validity of the assertion for a fixed r, to test $(r+1)$-hyper-reachability we first form intermediate candidates exactly as above. The only difference is that we are now dealing with candidates for r-hyper-reachability rather than for super-reachability.

Theorem 11 *A vector B is r-hyper-reachable iff it is r-hyper-reachable for a chain of modular multiplications, where each multiplier and modulus is less than* $(\max B)^{3^r}$.

Proof. We replace the upper bound $2\max B(1+\max B)$ in Theorem 4 by the much ruder upper bound $(\max B)^3$. Theorem 11 now follows because always the components of the vectors are smaller than the modulus.

Theorem 12 *Assume that B is r-hyper-reachable. Then the smallest m such that the r-hyper-reachability of B can be shown using only moduli $\leq m$ is effectively computable.*

Proof. One can use either the algorithm described in Theorem 10 or the more simply stated but less efficient algorithm due to Theorem 11. □

The part of Theorem 7 dealing with multipliers cannot be generalized by this technique. This is due to the fact that Lemma 4 cannot be applied in case of r-hyper-reachability. It is conceivable that a smaller t will work with a big increase in m.

It is clear that, for a fixed r, the algorithms due to Theorems 10-12 work in polynomial time (with respect to max B), where the degree of the polynomial depends on r.

Our final result is an immediate consequence of Theorem 10.

Theorem 13 *If B is known to be hyper-reachable, then the smallest r such that B is r-hyper-reachable can be effectively computed.*

9 Conclusion

The techniques developed here seem to be applicable to a great variety of topics dealing with knapsacks. We have mentioned above many open problems. In our estimation, the following are the most important among them. (i) Present criteria, other than Theorem 1, for constructing classes of vectors that are not super-reachable (resp. not r-hyper-reachable, not hyper-reachable). (ii) Do the r-hyper-reachable vectors form a strictly increasing hierarchy? (iii) Decidability of hyper-reachability?

10 References

[1] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsack. *IEEE Transactions on Information Theory* IT-24 (1978) 525-530.
[2] A. Salomaa. *Computation and Automata.* Cambridge University Press (1985).
[3] A. Salomaa. Knapsacks and superdogs. Formal Language Theory Column in *EATCS Bulletin* (June 1989).
[4] A. Salomaa. A cubic-time deterministic algorithm for modular knapsack problems. To appear in *Theoretical Computer Science.*
[5] A. Shamir. A polynomial time algorithm for breaking the basic Merkle - Hellman cryptosystem. *Proceedings of the 3rd FOCS Symposium* (1982) 145-152.

(Received October 23, 1989)

Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1990

Preserving two-tuple dependencies under projection

B. Thalheim* S. Al- Fedhagi†

Abstract

In relational databases, the semantics is modelled by dependencies defined for the whole set of attributes U and the relations on U. Given a set of dependencies on U and a relation R defined on a subset X of U, does there exist an extension of R to a relation R' on U which satisfies the set of dependencies. This problem is analyzed and a general solution is given in the context of two-tuple constraints.

1 Introduction

To achieve a certain level of scientific treatment of its subject-matter, many proposed descriptions of database models adopt different types of symbolic notations. Typically symbols of set theory, formal logic, graph theory, algebra, etc. are utilized to build a semi-formal language to express the main concepts of the subject matter. The relational model is the most obvious example of these models. It is typically described as having "mathematical elegance" and this mathematical characteristic is mentioned as one of its main advantages.

Dependency theory is a sub-field of the theory of relational database [1] that deals with formalizing integrity constraints and studying their mathematical structures. The importance of the theory stems from its implication on the design of the relational database. Dependency theory started with the very known inference rules for functional dependencies called Armstrong axioms and has grown into a very considerable field in the last fifteen years (see for instance [2], [3], [4]).

The influence of dependency theory and normalization theory, in general, on the database design process is definite. The rigorous treatment of the design process followed by the dependency theory is the sole attempt in that direction. Even though its full practicality is still to be proven, dependency theory forms the "showcase" to the claim that the database design field lends itself to formal treatment. Other attempts to formalize the database design process reflect types of "rules-off-thumb" and involve art more than science.

This paper deals with a type of dependency, called propositional dependencies, that are slightly more general than functional dependencies but still weaker than join dependencies. All issues that are related to functional dependencies can be

*Department of Mathematics, Faculty of Science, Kuwait University, KUWAIT

†Department of Electrical & Computer Engineering, Faculty of Engineering and Petroleum, Kuwait University, KUWAIT

analyzed on a better formal ground in the context of propositional dependencies. This is a major motivation for introducing propositional dependencies since there is still a great deal of interest in functional dependencies. Section three introduces few samples of the benefit of injecting propositional dependencies in the relational database issues.

Furthermore, propositional dependencies have their own significance since they form a constraint language. The language is rich as such that it encompasses all two-tuple constraints, still propositional dependencies are simple to identify, understand, and manipulate.

Additionally, propositional dependencies are interesting mathematical objects on their own. The study of their properties and their relationship to several technical issues, e.g., losslessness, may prove to be beneficial in the future.

The primary goal of this work is to show that propositional dependencies can be used to develop better solutions to the problems related to the issue of projected two-tuple constraints. The given set of propositional dependencies are converted in a standard disjunctive normal form and represented as "constraints tableau". The constraints tableaux, can be treated as ordinary relations, hence, different relational operations such as join and project can be applied to these tableaux. Certain modifications to the join and project operations are necessary since the operands are now constraints and not instances. In this paper the following problems are analyzed:

1. Given a set of functional dependencies Σ over the universal set of attributes U, and a relation $S(X), X \subseteq U$ such that S satisfies $\pi_X(\Sigma)$, i.e. the projection of Σ over X, then under what conditions does there exist a relation $R(U)$ that satisfies Σ such that $S = \pi_X(R)$, i.e. S is the projection of R.

2. Since functional dependencies are not preserved under projection, then under what conditions can we have a constraint preserving projection.

These problems are important in the design of the relational database. Let I be the set of all possible relations that satisfies Σ. A desired property of database schemes is that whenever the projection set $\pi_X(I)$ satisfies "projected constraints" it follows that I satisfies the constraints Σ. These problems are connected with three other well known database problems [5].

A. The extensibility problem: Let Σ be a set of integrity constraints on U and let R be a relation which satisfies Σ. Suppose that U' is an extension of U and Σ' is a set of some additional restrictions. Does there exist an extension of R on U' which satisfies both Σ and Σ' ? This problem was considered in the context of weak instances and the realization of the universal relation assumption [6]. Problem 1 gives a partial solution to this problem.

B. The view update problem: Given a conceptual scheme (U, Σ) and a set of views $(U_1, \Sigma_1) \ldots, (U_n, \Sigma_n)$, let $R_1, \ldots, R_n$ be relations for these views. Does there exist an algorithm to decide whether an update of one relation is consistent with the other relations?

C. The implied constraint problem: Let $S_1 = (U_1, \Sigma_2)$ and $S_2 = (U_2, \Sigma_2)$ be two conceptual schemes. Consider the database mapping $\gamma : S_1 \longrightarrow S_2$. Such a mapping induces in a natural way a mapping $\gamma^* : \mathrm{SAT}(S_1) \longrightarrow \mathrm{SAT}\ (S_2)$ where $\mathrm{SAT}(S_i)$ denotes the set of all relations on U_i satisfying Σ_i. We ask whether γ is correct, i.e. $\gamma^*(\mathrm{SAT}(S_1)) = \mathrm{SAT}(S_2)$. In general, this problem is undecidable [7]. A solution is known only for some special cases. Problem 2 can contribute to a general solution of this problem.

Known solutions to problem 1 mentioned previously above require that $S(X)$ should satisfy additional non-two-tuple constraints in order to guarantee the existence of $R(U)$. Our solution characterizes the conditions under which there exists a relation $R(U)$, utilizing only the given two-tuple constraints such as functional dependencies or propositional dependencies. Similarly, we characterize conditions under which we can have a constraint preserving projection, utilizing only two-tuple constraints.

We assume that reader is familiar with relational database theory, and with some background in propositional logic. U is used to denote the set of attributes of the universal relation. X, Y, Z, W (possibly subscripted) are used to denote relation schemes. $S(X), R(X)$ are used to denote relations instances over $X \subseteq U$. The relations $S(X)$ and $R(X)$ may be written as S and R, respectively, when X is understood or immaterial. Small letters u and w are used to denote tuples in relation instances (e.g. $u_1 \in R$). Let us denote by $\underline{R}$ (or $\underline{R}(U)$) the set of all relations on U. The projection of R to a subset X of U is denoted by $R[X]$, i.e. $R[X] = \{u[X] | u \in R\}$ where $u[X]$ is the restriction of u to X.

A relation R satisfies a set of constraints Σ if it satisfies each constraint in Σ. $\mathrm{SAT}(X, \Sigma), X \subseteq U$, is the set of all relations over X that satisfy Σ. $\mathrm{SAT}(X, \Sigma)$ may be written as SAT (Σ) or $\mathrm{SAT}(X)$ when Σ or X are understood respectively.

2 Propositional Dependencies (PDs)

Propositional dependencies form a formal apparatus to express constraints on two-tuples relations. They provide a foundation based on propositional calculus that is suitable for this purpose.

Let $U = \{A_1, \ldots, A_n\}$ be the given set of attributes. With each attribute A there is associated a *propositional variable* A'. For two different tuples t, t' on U, the propositional variable A' denotes the proposition: "The two tuples agree in the A-value". The negation of $A', -A'$, denotes the contrary, that these tuples have different A-values. Without any loss of generality we denote by A the attribute and the propositional variable.

Given the set $\{\wedge, \vee, -, \rightarrow, \leftrightarrow\}$ of logical connectives (conjunction, disjunction, negation, implication, equivalence) and the set U, the set $L(U)$ of *propositions* on U is defined as follows:

1. Any propositional variable is a proposition.
2. If H and H' are propositions then $-H, (H \wedge H'), (H \vee H'), (H \rightarrow H'), (H \leftrightarrow H')$ are propositions.

For any pair of different tuples (t, t') and the set $L(U)$ we define an *interpretation* of propositions as follows:

1. The propositional variable A is true for (t, t'), if $t[A] = t'[A]$ and otherwise A is false.
2. $-H$ is valid for (t, t') if H is false; furthermore, for (t, t'): $H \wedge H')$ is said to be valid for (t, t') if H and H' are valid for (t, t') (if "H and H'"); analogously the validity of $(-H \vee H')$ is defined by "H or H'", $(H \rightarrow H')$ by $(-H \vee H')$ and $(H \leftrightarrow H')$ by $((H' \longrightarrow H) \wedge (H \longrightarrow H'))$.

The validity of H for different t, t' is denoted by ${}_{(t,t')}| = H$.

For a set of attributes $X = \{B_1, \ldots, B_m)$ the set X is also used to denote the proposition $B_1 \wedge \ldots \wedge B_m$.

The notation ${}_{(t,t')}| = H$ can be extended to ${}_R| = H$ as follows:
The proposition H is valid in the relation R (denoted by ${}_R| = H$) iff for any pair of different tuples (t, t') from R, H is valid; i.e. ${}_{(t,t')}| = H$.

A set $\underline{H}$ of propositional dependencies is valid in R (denoted by ${}_R| = \underline{H}$) if all elements of $\underline{H}$ is valid in R. Note that we will use the under bar notation whenever sets of relations or formulae are to be denoted.

For a subset $\underline{R}'$ of $\underline{R}$, a given set $\underline{H}$ of propositional dependencies and a propositional dependency H we say that the set $\underline{H}$ implies H in $\underline{R}$ if for any relation R from $\underline{R}'$ in which $\underline{H}$ is valid, it holds also ${}_R| = H$ (denoted by $\underline{H}_{\underline{R}'}| = H$ or by $\underline{H}| = H$ for $\underline{R}' \subseteq \underline{R}$).

Corollary 1 *For any relation R with $|R| \leq 1$ and any proposition in $L(U)$; $\underline{H}_R| = H$.*

The "world of two tuple relations" [8] $\underline{R}_2$ denotes the set of two-tuple relations that can be constructed from possible relations with two or more tuples. A two-tuple constraint is a condition that is imposed in the world of two-tuple relations. For example, the proposition $H = ((-X \wedge Y) \vee (X \wedge -Y))$ expresses the following constraint: for any two different tuples t, t', ${}_{(t,t')}| = H$ iff the two tuples differ in the X-value and match in the Y-value or they match in the X-value and they differ in the Y-value. Functional dependencies are examples of two-tuple constraints.

Any formula from $L(U)$ is called *propositional dependency.*

Corollary 2 *For any set $\underline{R}'$ which contains $\underline{R}_2$, any set $\underline{H}$ of propositional dependencies and a propositional dependency H the following are equivalent:*
(I) $\underline{H}| = H$.
(II) $\underline{H}_{\underline{R}'}| = H$.

Example 1 [1]. Let $U = \{A, B, C, D, E\}, \Sigma = \{A \rightarrow E, B \rightarrow E, CE \rightarrow D\}$. Then the following propositional dependencies are equivalent to the dependencies in Σ:

$-A \vee E$
$-B \vee E$
$-(CE) \vee D$

Σ implies that AC is a key of any relation satisfying Σ. This property is expressed by the propositional dependency $-(AC)$.

Example 2. Suppose that $XY = U$ where XY denotes the union of the sets X and Y. For a functional dependency $X \rightarrow Y$, e.g. X is the *key* of U, the equivalent propositional dependency is $-X$. That is, for any two tuples in the relations on U, the two tuples differ in the X-value.

Example 2 shows two propositional formulae that have the same meaning on a given universe U because of the definition of the interpretation of H and the formula $H \wedge -U$. The disjunct $-U = (-A_1 \vee \ldots \vee -A_n)$ for $U = \{A_1, \ldots, A_n\}$ is always assumed because relations are defined to be sets and two tuples of a relation should be different. Therefore, the disjunct $-U$ can be eliminated in all propositional dependencies or can be added to all propositional dependencies. Instead of considering the whole propositional logic $L(U)$ we add to all dependency sets H the axiom $(-A_1 \vee \ldots \vee -A_n)$ as an axiom in our propositional logic called dependency propositional logic, DPL.

Delobel and Casey [9] were the first to relate the functional dependencies to material implications in the two-valued Boolean algebra which is equivalent to propositional logic. In [17], Demetrovics et al. considered the extension of functional dependencies to different classes of Boolean dependencies. Using the theory of Boolean functions, there can be derived different algorithms for scheme design [2]. Propositional dependencies were first introduced by Sagiv et al. as "Boolean dependencies" [10]. They were studied in details, independently by Thalheim [12], Al-Fedaghi [11] and Berman and Blok [13]. In [10] it is claimed that the consequence relation for the class of Boolean dependencies is equivalent to the consequence relation for propositional logic. Unfortunately this is not true because H is a consequence of $(H \wedge -U)$ but $(H \wedge -U)$ is not a consequence of H. We notice that the idea of propositional dependencies is basically a dependency system which can replace the formal system of functional dependencies. There is a set of propositional dependencies that is equivalent to any given set of functional dependencies but not vice versa. For example, the formula $(A \rightarrow (B \vee C))$ is not equivalent to any set of functional dependencies. Therefore the family of propositional dependencies has more expressive power. Furthermore the simplicity of the propositional calculus makes the propositional dependencies a very practical tool.

3 The Propositional Constraints Tabelaux

A formula G is said to be in the disjunctive normal form if G has the form of $G_1 \vee G_2 \vee \ldots \vee G_m, m \geq 1$, where each $G_i, 1 \leq i \leq m$, is a conjunction of literals. A standard disjunctive normal form (SDNF) is a disjunctive normal form where each conjunction contains all propositional variables.

Any set Σ of propositional dependencies corresponds to a unique propositional logic formula in the standard disjunctive normal form. It is sometimes very convenient to work with these standard forms instead of Σ. Several issues such as "equivalence" can be easily analyzed through studying standard forms. A constraint tableau, c-tableau, Σ is a $0-1$ matrix that corresponds to the disjunctive normal form of Σ. This tableau is denoted by $T(\Sigma)$ or T when the set Σ is understood.

Let $\Sigma_N = (C_1 \vee C_2 \vee \ldots \vee C_k)$ be the SDNF of $\Sigma(U)$. Each conjunct C_i includes $m = |U|$ literals. The tableau $T(\Sigma)$ is the $0-1$ matrix where row i corresponds to C_i and column j corresponds to attribute (i.e. propositional variable) $A_j \in U$. The entry (i, j) of T is defined as follows:

$$(i, j) = \begin{cases} 1 & \text{if } A_j \text{ is in } C_i \\ 0 & \text{if } -A_j \text{ is in } C_i \end{cases}$$

Example 3: The set of functional dependencies Σ, given in Example 1 can be represented by the c- tableau of Figure 1.

We use the names of attributes to denote the columns of the c-tableau. Since any row of the c-tableau represents a conjunction of literals in the SDNF, it makes sense to say that the two-tuple relation R satisfies that row. In general, we say that a given relation R satisfies that row or it satisfies a given c-tableau T. Furthermore, SAT(Σ) may be denoted by SAT($T(\Sigma)$) or SAT (T) when Σ is understood.

Definition: Let T be a c-tableau and $\{A_1, \ldots, A_n\}$ be the set of its attributes (i.e. the columns names). Without loss of generality, the projection of T over $X = \{A_1, \ldots, A_m\}, m < n$, is defined as follows:

$$\overline{\pi}_X(T) = \{t | t \quad \text{is the subrow of } T \text{ over the attributes } X \text{ such that}$$

$$(u[A_1] = 0) \vee (u[A_2] = 0) \vee \ldots \vee (u[A_m] = 0)\}$$

That is, the all 1's sub-row is dropped out of $\overline{\pi}_X(T)$.

A	B	C	D	E
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1

Figure 1 : $T(\Sigma)$ of dependenciesin example 2

Example 3. (continued). For the tableau presented in Figure 1 the following projections are defined.

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	0	1

$\overline{\pi}_{ABCD}(T)$

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0

$\overline{\pi}_{ABC}(T)$

An Ω-tableau $\Omega(X)$ is defined as the two-tuple $< \hat{T}(X), R^2(X) >$ where $\hat{T}(X)$ is a 0-1 matrix and $R^2(X)$ is a set of pairs of labels $\{(u_i, u_j), i < j\}$. An Ω- tableau may correspond to a relation $R(X)$ as follows. The set $R^2(X)$ represents all pairs of tuples of $R(X)$. Hence $|R^2(X)| = (\frac{n}{2}), n = |R(X)|$. A mapping μ is defined between the pairs $\{(u_i, u_j)\}, i \neq j$, in $R^2(X)$ and the rows in $\hat{T}(X)$ as follows:

The A_k-value in row $\mu(u_i, u_j)$ of

$$\hat{T}(X) = \begin{cases} 1 & \text{if } u_i[A_k] = u_j[A_k] \\ 0 & \text{otherwise} \end{cases}$$

Tuple	A_1	A_2
u_1	a_1	b_1
u_2	a_1	b_2
u_3	a_2	b_2
u_4	a_2	b_3

(a) $R(A_1A_2)$

$\hat{T}(A_1 A_2)$	
$A_1\ A_2$	$R^2(A_1\ A_2)$
1 0	u_1, u_2
0 0	u_1, u_3
0 0	u_1, u_4
0 1	u_2, u_3
0 0	u_2, u_4
1 0	u_3, u_4

(b) $\Omega(A_1A_2)$

Figure 2 : Relation and its Ω – tableau.

for all $A_k \in X$.

Example 4: Figure 2 shows a given relation and its corresponding Ω-tableau.

Now we formalize this approach.

For any c-tableau T and a relation R on $U = \{A_1, \ldots, A_n\}$ that satisfies T we can define another extended tableau as follows. Let σ be the following function:

$$\sigma(u, w) = \begin{cases} 1 & \text{if } u = w \\ 0 & \text{if } u \neq w \end{cases}$$

where u, w are values from the domains of R. Then we define a set of rows as follows:

$$\Omega(R) = \{(\sigma_1, \ldots, \sigma_n, b_1, b_2) | \sigma_i = \sigma(b_1[A_i], b_2[A_i])\}$$

where b_1, b_2 are labels of tuples of R.

Notice that for $T(R) = \{(\sigma_1, \ldots, \sigma_n) | (\sigma_1, \ldots \sigma_n, b_1, b_2) \in \Omega(R) \text{ for } b_1, b_2 \in R\}$ and the c-tableau T we get $T(R) \leq T$. Furthermore the projection of $\Omega(R)$ can be formalized as follows:

$\pi_X(\Omega(R)) = \{t | t$ is the projection of a row of $\Omega(R)$ on X leaving the labels out $\}$.

Example 5. Consider the following relations R_1, R_2 defined on $U = \{A, B, C, D\}$ and their Ω-tableaus.

R_1:	A	B	C	D
u_1	1	3	5	7
u_2	2	4	5	8
u_3	1	4	6	8

$\Omega(R_1)$:	A	B	C	D	t_1	t_2
	0	0	1	0	u_1	u_2
	1	0	0	0	u_1	u_3
	0	1	0	1	u_2	u_3

R_2:	A	B	C	D
u_1	1	3	5	8
u_2	2	4	5	8
u_3	1	4	6	8

$\Omega(R_2)$:	A	B	C	D	t_1	t_2
	0	0	1	1	u_1	u_2
	0	1	0	1	u_2	u_3
	1	0	0	1	u_1	u_3

We can define arbitrary tableaus in the following way. Given a finite abstract set of tuple names $\{u_1, \ldots, u_m\}$ and a domain set $\{A_1, \ldots, A_n\}$. Then any set

$$\{(\sigma_1, \ldots, \sigma_n, u_i, u_j) | 1 \leq i < j < m, \sigma_k \in \{0, 1, \}\}$$

form an Ω-tableau.

Example 6. Let $u = \{A, B\}$. Consider the following Ω-tableau

Ω	A	B	t_1	t_2
	0	0	u_1	u_2
	1	0	u_1	u_3
	1	0	u_2	u_3

There exists no relation R with $\Omega = \Omega(R)$. To prove this, we show that if R exists then the tuples u_1 and u_2 should be equal on the attribute A. Since $u_1[A] = u_3[A]$ and $u_3[A] = u_2[A]$ then $u_1[A] = u_2[A]$. Clearly, this contradicts the Ω-tableau where $u_1[A] \neq u_2[A]$.

The tableau Ω is said to be realizable if there exists a relation R with $\Omega = \Omega(R)$. Let us first consider the realizability of Ω-tableaus. We define a relation for any attribute $A \in U = \{A_1, \ldots, A_4\}$ as follows:

$\varsigma_{A_i,\Omega}\{(u_1, u_2)|$ (such that there is a row

$$\sigma_1, \sigma_2, \ldots, \sigma_n, u_3, u_4) \in \Omega \quad \text{with} \quad \sigma_i = 1, \quad \text{and}$$

$$((u_3 = u_1, u_4 = u_2) \quad \text{or} \quad (u_3 = u_2, u_4 = u_1)) \quad \text{or} \quad u_1 = u_2\}$$

Theorem 3 *The tableau Ω is realizable iff for all A in U $\varsigma_{A,\Omega}$ is an equivalence relation.*

Proof.

1. For some A in U let $\varsigma_{A,\Omega}$ be not an equivalence relation. Since $\varsigma_{A_i,\Omega}$ is reflexive and symmetric then there exists in Ω three tuples $(\sigma_1^1, \ldots, \sigma_n^1, v_1, v_2), (\sigma_1^2, \ldots, \sigma_n^2, v_2, v_3)(\sigma_1^3, \ldots, \sigma_n^3, v_1, v_3)$ such that $\sigma_i^1 = \sigma_i^2 = 1$ and $\sigma_i^3 = 0$. Consequently, we get $v_1[A_i] = v_2[A_i] = v_3[A_1]$, and $v_1[A_i] = v_2[A_i]$, i.e. a contradiction. Therefore, Ω is not realizable.

2. If for all A in U $\varsigma_{A,\Omega}$ is an equivalence relation and $\{u_1, \ldots, u_m\}$ is the set of abstract tuples used in Ω then we can define a relation $R = \{u_1, \ldots, u_m\}$ using the partitions $P_{A,\Omega}$ defined by $\varsigma_{A,\Omega}$. For $P_{A,\Omega} = \{V_1, \ldots, V_k\}$ where V_i is an equivalence class we define $u_i[A] = j$ iff $v_i \in V_j$. Obviously $\Omega(R)$ is equal to Ω.

For equivalence relations ς_1, ς_2 on $R = \{u_1, \ldots, u_m\}$, the following operations are defined: $\varsigma_1 \wedge \varsigma_2$ (intersection), $\varsigma_1 + \varsigma_2$ (the smallest equivalence relation containing ς_1 and ς_2), and the comparison $\varsigma_1 \leq \varsigma_2(\varsigma_1 \leq \varsigma_2$ iff $(u_1, u_2) \in \varsigma_1$ implies that $(u_1, u_2) \in \varsigma_2)$.

Corollary 4 *Let R be a relation on $U = \{A_1, \ldots, A_n\}$, and $X = \{B_1, \ldots, B_m\}, Y = \{C_1, \ldots C_k\}, Z \subseteq \{D_1, \ldots, D_l\} \subseteq U$. Furthermore assume the Ω-tableau $\Omega(R)$, and the equivalence relations $\varsigma_{A,\Omega(R)}$ for $A \in U$. Then:*

i) $X \to Y$ is valid in R iff $\varsigma_{B_{1,\Omega(R)}} \wedge \ldots \wedge \varsigma_{B_{m,\Omega(R)}} \leq \varsigma_{C_{i,\Omega(R)}}$ for all $i, 1 \leq i \leq k$.

ii) $X \to Y, Z \to Y$ is valid in R iff for all $i, 1 \leq i \leq k$ $(\varsigma_{B_{1,\Omega(R)}} \wedge \ldots \wedge \varsigma_{B_{m,\Omega(R)}}) + \varsigma_{D_{1,\Omega(R)}} \wedge \ldots \wedge \varsigma_{D_{l,\Omega(R)}}) \leq \varsigma_{C_i,\Omega(R)}$

Proof: Suppose that $X \to Y$ is valid in R. Consequently for two tuples u_1, u_2 in R if $u_1[X] = u_2[X]$ then $u_1[Y] = u_2[Y_2]$. Hence, $(u_1, u_2) \in (\varsigma_{B_{1,\Omega(R)}} \wedge \ldots \wedge \varsigma_{B_{m,\Omega(R)}})$ and $(u_1, u_2) \in \varsigma_{C_{i,\Omega(R)}}$ for any $C_i \in Y$.

In $\Omega(R)$ the property $\varsigma_{B_{1,\Omega(R)}} \wedge \ldots \wedge \varsigma_{B_{m,\Omega(R)}} \leq \varsigma_{C,\Omega(R)}$ is easy to check. If for a row $(\sigma_1, \ldots, \sigma_n, u, v)$ in $\Omega(R), \sigma_i = 1$ for all $A_i \in X$ then for any $A_j \in Y$ $\sigma_j = 1$.

Example 7: Consider the relations R_1 and R_2 given in example 5. The relations $R_1, R_2 \in$ SAT $(\{AC \to D, BC \to D\})$. The dependencies $A \to D, B \to$

$D, C \rightarrow D$ are also valid in R_2 whereas only $B \rightarrow D$ is valid in R_1. Furthermore $D \rightarrow B$ is valid in R_1 but $D \rightarrow B$ is invalid in R_2.

1) Let $\Omega = \Omega(R_1)$. The relationships

$$\varsigma_{A,\Omega} \not\leq \varsigma_{B,\Omega}, \varsigma_{A,\Omega} \not\leq \varsigma_{C,\Omega}, \varsigma_{A,\Omega} \not\leq \varsigma_{D,\Omega},$$
$$\varsigma_{B,\Omega} \not\leq \varsigma_{A,\Omega}, \varsigma_{B,\Omega} \not\leq \varsigma_{C,\Omega}, \varsigma_{B,\Omega} \leq \varsigma_{D,\Omega},$$
$$\varsigma_{C,\Omega} \not\leq \varsigma_{A,\Omega}, \varsigma_{C,\Omega} \not\leq \varsigma_{B,\Omega}, \varsigma_{C,\Omega} \not\leq \varsigma_{D,\Omega},$$
$$\varsigma_{D,\Omega} \not\leq \varsigma_{A,\Omega}, \varsigma_{D,\Omega} \leq \varsigma_{B,\Omega}, \varsigma_{D,\Omega} \not\leq \varsigma_{C,\Omega}$$

can be represented by

$\Omega, \leq$	$\varsigma_{A,\Omega}$	$\varsigma_{B,\Omega}$	$\varsigma_{C,\Omega}$	$\varsigma_{D,\Omega}$
$\varsigma_{A,\Omega}$	1	0	0	0
$\varsigma_{B,\Omega}$	0	1	0	1
$\varsigma_{C,\Omega}$	0	0	1	0
$\varsigma_{D,\Omega}$	0	1	0	1

2) Let $\Omega = \Omega(R_2)$. We get the following table

$\Omega, \leq$	$\varsigma_{A,\Omega}$	$\varsigma_{B,\Omega}$	$\varsigma_{C,\Omega}$	$\varsigma_{D,\Omega}$
$\varsigma_{A,\Omega}$	1	0	0	1
$\varsigma_{B,\Omega}$	0	1	0	1
$\varsigma_{C,\Omega}$	0	0	1	1
$\varsigma_{D,\Omega}$	0	0	0	1

Notice that corollary 4 can be extended to propositional dependencies. For instance, the dependency $D \rightarrow A \vee B \vee C$ is valid in R_2. Generally, the dependency $X \rightarrow Y_1 \vee \ldots \vee Y_k$ is valid in R iff for $\Omega = \Omega(R)$

$$\bigcap_{A \in X} \varsigma_{A,\Omega} \leq \cup_{i=1}^{k} \big(\bigcap_{B \in Y_i} \varsigma_{B,\Omega} \big).$$

where $\cup$ denotes the union of sets, i.e. for sets ς_1, ς_2 of sets $\varsigma_1 \cup \varsigma_2 = \{V \mid$ there are $V_1 \in \varsigma_1, V_2 \in \varsigma_2 : V = V_1 \cup V_2\}$.

4 Projections of Constraints and Relations

According to Maier [1] the notion of "projected constraints" is well defined for functional and multivalued dependencies. If $W \subseteq U$, and Σ is a set of functional and multivalued dependencies then $\pi_W(\Sigma)$ consists of those $X \rightarrow Y$ and $X \rightarrow\rightarrow Y$ such that:

i) there is some $X \rightarrow Z$ or $X \rightarrow\rightarrow Z$ in Σ^+ where Σ^+ is the closure of Σ,
ii) $X \subseteq W$, and
iii) $Y = Z \cap W$.

For functional dependencies, it is always assumed that $Y = Z$. Hence, for the given set of attributes U and set of functional dependencies (over U), $\pi_W(\Sigma) = \{X \rightarrow Y$ in $\Sigma^+ XY \subseteq W\}$ where Σ^+ is the closure of Σ.

Example 8. Consider the sets $U = \{A, B, C, D, E\}, \Sigma = \{A \rightarrow E, B \rightarrow E, CE \rightarrow D\}$ of example 1. Then for $X = \{A, B, C, D\}, \pi_x(\Sigma) = \{AC \rightarrow D, BC \rightarrow D\}$.

In a similar way, the projections of disjunctive normal forms and of c-tableaus are defined.

Given a standard disjunctive normal form $\Sigma_N = C_1 \vee C_2 \vee \ldots \vee C_k$ of propositional dependencies over U. If $X \subseteq U$, then the projection of Σ_N over X is defined as the propositional dependency

$$\pi_x(\Sigma_N) = (C'_1 \vee C'_2 \vee \ldots \vee C'_k) \wedge (-A_1 \vee -A_2 \vee \ldots \vee -A_l)$$

where $X = \{A_1, A_2, \ldots A_l\} \subseteq U$ and C'_i is the disjunct produced from C_i after removing all propositional variables not in X.

Example 9. The standard disjunctive normal form of Σ is
$\Sigma_N = -A-B-C-D-E \vee -A-B-C-DE \vee -A-B-CD-E \vee -A-B-CDE$
$\vee - A - BC - D - E \vee -A - BCD - E \vee -A - BCDE$
$\vee - AB - C - DE \vee -AB - CDE$
$\vee - ABCDE \vee A - B - C - DE \vee A - BCDE \vee AB - CDE$
for $X = \{A, B, C, D\}$.
Then $\pi_x(\Sigma_N) = (-A - B - C - D \vee -A - B - CD \vee -A - BC - D$
$\vee - A - BCD \vee -AB - C - D \vee -AB - CD \vee -ABCD$
$\vee A - B - C - D \vee A - BCD \vee AB - CD) \wedge (-A \vee -B \vee -C \vee D)$.

Now let us introduce the extension of Ω-tableaux. Given the sets

$$X = \{A_1, \ldots, A_n\}, Y = \{A_1, \ldots, A_n, B_1, \ldots, B_k\}$$

and a set

$$\Omega = \{(\sigma_1, \ldots, \sigma_n, v, w)\}.$$

A tableau $\Omega' = \{(\sigma_1, \ldots, \sigma_n, \sigma_1, \ldots, \sigma_k, v, w)\}$ defined on Y is said to be an extension of Ω to Y if $\pi_X(\Omega') = \Omega$.

For realizable Ω-tableaux we introduce the set $\Psi_Y = \Psi_Y(\Omega) = \{\Omega' | \Omega'$ is an extension of Ω to $Y.\}$

Example 10. Given $X = \{A, B, C, D\}, Y = \{A, B, C, D, E\}$. The following set of Ω'-tableaux are extensions of $\Omega_1 = \Omega(R_1)$ to Y where R_1 is given in example 5.

Ω'_{11}	A	B	C	D	E	t_1	t_2
	0	0	1	0	0	u_1	u_2
	0	1	0	1	0	u_2	u_3
	1	0	0	0	0	u_1	u_3

Ω'_{12}	A	B	C	D	E	t_1	t_2
	0	0	1	0	1	u_1	u_2
	0	1	0	1	0	u_2	u_3
	1	0	0	0	0	u_1	u_3

Ω'_{13}	A	B	C	D	E	t_1	t_2
	0	0	1	0	0	u_1	u_2
	0	1	0	1	1	u_2	u_3
	1	0	0	0	0	u_1	u_3

Ω'_{14}	A	B	C	D	E	t_1	t_2
	0	0	1	0	0	u_1	u_2
	0	1	0	1	0	u_2	u_3
	1	0	0	0	1	u_1	u_3

Ω'_{15}	A	B	C	D	E	t_1	t_2
	0	0	1	0	1	u_1	u_2
	0	1	0	1	1	u_2	u_3
	1	0	0	0	1	u_1	u_3

Example 11. Given $X = \{A, B, C, D\}, Y = \{A, B, C, D, E\}$ and $\Omega_1 = \Omega(R_2)$ where the relation R_2 is given in example 5. The set $\Psi_Y = \Psi_Y(\Omega_2)$ can be represented by the following table

Ω'_n	$ABCD$	$t_1 t_2$	$\Omega'_{21} : E$	$\Omega'_{22} : E$	$\Omega'_{23} : E$	$\Omega'_{24} : E$	$\Omega'_{25} : E$
	0 0 1 1	$u_1\, u_2$	0	1	0	0	1
	0 1 0 1	$u_2\, u_3$	0	0	1	0	1
	1 0 0 1	$u_1\, u_3$	0	0	0	1	1

The first problem presented in section 1 can now be stated as follows: Given a set Σ of integrity constraints defined on U and a relation $R \in$ SAT $(\pi_x(\Sigma))$ where R is defined on X, then does there exist a relation $R' \in$ SAT (Σ) such that $R = \pi_x(R')$? This problem is equivalent to the following problem: Given a relation $R \in$ SAT $(\pi_x(\Sigma))$ where R is defined on X and Σ is defined on U. Does there exist a relation $R' \in$ SAT (Σ) such that $\Omega(R') \in \Psi_U(\Omega(R))$?
For a given set Σ of functional dependencies let Eq (Σ) denote the set of relationships defined in corollary 4.

Example 12. Consider Σ and Σ_X given in example 8.

$$\text{Eq}(\{A \rightarrow E, B \rightarrow E, CE \rightarrow D\}) = \{\varsigma_A \leq \varsigma_E, \varsigma_B \leq \varsigma_E, \varsigma_C \wedge \varsigma_E, \leq \varsigma_D\}$$

$$\text{Eq}(\{AC \rightarrow D, BC \rightarrow D\}) = \{\varsigma_A \wedge \varsigma_C \leq \varsigma_D, \varsigma_B \wedge \varsigma_C, \leq \varsigma_D\}.$$

As a corollary of Theorem 3 and corollary 4 we get directly the solution of the last problem.

Theorem 5 *Let Σ be a set of functional dependencies defined on X, and let X be a subset of U. For a relation R defined on X such that R satisfies $\pi_X(\Sigma)$ there exists an extension R' in SAT (Σ) if and only if there is in $\Psi_U(\Omega(R))$ a set Ω' such that the relationships of $Eq(\Sigma)$ are fulfilled in Ω'.*

Example 13. Let us continue examples 5, 8, 10, 11, 12.
The relationship $\varsigma_A \leq \varsigma_E$ from Eq (Σ) is violated in $\Omega'_{11}, \Omega'_{12}, \Omega'_{13}$.$\Omega'_{14}$ violates $\varsigma_B \leq \varsigma_E, \Omega'_{15}$ violates $\varsigma_C \wedge \varsigma_E \leq \varsigma_D$. Therefore there does not exist any relation extending relation $R_1 \in$ SAT $(\pi_x(\Sigma))$ which satisfies Σ. The relationships $\varsigma_A \leq \varsigma_E$ and $\varsigma_B \leq \varsigma_E$ are not valid in $\Omega'_{21}, \Omega'_{22}, \Omega'_{23}, \Omega'_{24}$. The set Eq (Σ) is valid for Ω'_{25}. Therefore there exists a relation R' in SAT (Σ)-with $R = \pi_x(R')$. An example of such a relation is the following relation R_3

R_3	A	B	C	D	E
u_1	1	3	5	8	9
u_2	2	4	5	8	9
u_3	1	4	6	8	9

Based on theorem 5, an algorithm can be developed for the computation of an Ω' tableau if there exists such a set for a given R, Σ, and $\Omega(R)$.

Algorithm.

Input. $U = \{A_1, \ldots, A_n\}, X = \{B_1, \ldots, B_k\} \subseteq U, \Sigma$ is a set of functional dependencies defined on U, and the relation R defined on X.

Output. A relation $R' \in$ SAT (Σ) with $R = \pi_X(R')$ if there exists such a relation.

(i) Construct $\pi_X(\Sigma)$.

(ii) Compute $\Omega(R)$.

(iii) Compute Eq (Σ), and Eq $(\pi_X(\Sigma))$

(iv) If $\Omega(R)$ violates Eq $(\pi_X(\Sigma))$ then output that there does not exist a relation R.

(v) Construction of Ψ tables.

1. If for some $Y \subseteq U - X, Z \subseteq X$, there is a dependency $Z \to Y \in \Sigma(Z = \{C_1, \ldots, C_m\}, Y = \{D_1, \ldots, D_p\})$ then copy the 1-entries in the columns of Z to all columns of Y. The result is the table Ω_1.
2. Compute the 1-entries according to theorem 3. (All columns in $U - X$ must be represented by equivalence relations). The result is Ω_2.
3. If for some $Y \subseteq U - X, Z \subseteq X$, there is a dependency $Z \to Y \in \Sigma(Z = \{C_1, \ldots, C_m\}, Y = \{D_1, \ldots, D_p\}$ and a row with 0-entry in one of the Z-columns and for alll Y-columns except one there are 1-entries in that row then write a 0 in the remaining Y-column in that row. The result is Ω_3.
4. If for some column in $U - X, u_i[A] = u_j[A]$ and $u_i[A] = u_k[A]$ then enter 0 in this column for the (u_j, u_k)-row (this is the closure for equivalence relations). The result is Ω_4.
5. If Ω_4 violates $Eq(\Sigma)$ then output that there can not exist such a relation R'.
6. Compute $\Psi(\Omega_4)$ and check against Eq (Σ). If $\Psi(\Omega_4)$ is empty then there is no relation R' satisfying the requirement. If $\Psi(\Omega_4)$ is not empty then use the proof of theorem 3 for the computation of R'.

Example 14 [15]. Given $U = \{A, B, C, E, F, G, H\}, \Sigma = \{A \to G, B \to G, C \to H, E \to H, GH \to F\}, X = \{A, B, C, E, F\}$, and the relation R:

R	A	B	C	E	F
u_1	1	1	1	1	1
u_2	2	2	2	2	2
u_3	3	3	1	2	3
u_4	1	2	3	3	4

We get after (i) in the algorithm:
$\pi_x(\Sigma) = \{AC \to F, AE \to F, BC \to F, BE \to F\}$, and after (ii) the Ω-tableau:

$\Omega(R)$	A	B	C	E	F	t_1	t_2
	0	0	0	0	0	u_1	u_2
	0	0	1	0	0	u_1	u_3
	0	0	0	1	0	u_2	u_3
	0	1	0	0	0	u_2	u_4
	0	0	0	0	0	u_3	u_4
	1	0	0	0	0	u_1	u_4

Since $\Omega(R)$ obeys $\varsigma_A \wedge \varsigma_C \leq \varsigma_F, \varsigma_A \wedge \varsigma_E \leq \varsigma_F, \varsigma_B \wedge \varsigma_C \leq \varsigma_F$, and $\varsigma_B \wedge \varsigma_E \leq \varsigma_F$ we continue with step (v) using Eq $(\Sigma) = \{\varsigma_A \leq \varsigma_G, \varsigma_B \leq \varsigma_G, \varsigma_C \leq \varsigma_H, \varsigma_E \leq \varsigma_H, \varsigma_G \wedge \varsigma_H \leq \varsigma_F\}$. The following table represents the step (v) of the algorithm:

$ABCEF\ t_1t_2$	after applying the first 4 relationships $\Omega_1 GH$	equivalence relations on $\Omega_1 : \Omega_2 GH$	applying $\zeta_G \wedge \zeta_H \leq \zeta_F$ to Ω_2 : $\Omega_4 G\ H$	check of Eq (Σ) in Ω_3
$00000\ u_1u_2$		1 1	1 1	contradiction
$00100\ u_1u_3$	1	1	0 1	
$10000\ u_1u_4$	1	1	1 0	
$00010\ u_2u_3$	1	1	0 1	
$01000\ u_2u_4$	1	1	1 0	
$00000\ u_3u_4$			0 0	

Therefore we conclude that there can not exist a relation R' in SAT (Σ) such that $R = \pi_x(R')$. Using our approach we get usually a set of contradictions or a set $\psi' \subseteq \psi(\Omega(R))$ of candidate Ω-tableaus for extensions of R.

Example 15. Let us continue example 13.
In step (v) we get for $\Omega_1 = \Omega(R_1)$, and $\mathrm{Eq}(\Sigma) = \{\zeta_A \leq \zeta_E, \zeta_B \leq \zeta_E, \zeta_C \wedge \zeta_E \leq \zeta_D\}$. The following table is the result of the application of the algorithm.

A	B	C	D	t_1	t_2	Ω_{11}	E	Ω_{12}	E	
0	0	1	0	u_1	u_2				1	contradiction
0	1	0	1	u_2	u_3		1		1	
1	0	0	0	u_1	u_2		1		1	

For $\Omega_2 = \Omega(R_2)$ and $Eq(\Sigma)$ we obtain

A	B	C	D	t_1	t_2	Ω_{21}	E	$\Omega_{22} = \Omega_{23} = \Omega_{24}$	E
0	0	1	1	u_1	u_2				1
0	1	0	1	u_2	u_3		1		1
1	0	0	1	u_1	u_4		1		1

In step (vii) we get the relation

R_3	A	B	C	D	E
	1	3	5	8	9
	2	4	5	8	9
	1	4	6	8	9

For the relations and the dependency sets used in examples 13 and 14, Fagin [14] defines the following "curious dependency" using the equality generality dependencies as an additional condition to guarantee the preservation of dependencis under projection:

if $u_1(B) = u_2(B), u_1(C) = u_3(C), u_2(A) = u_3(A)$ and $u_1(D) = u_2(D)$ then $u_1(D) = u_3(D)$.

This condition is not a sufficient condition as it is shown in the following example.

Example 16. Consider $U, \Sigma, X, \pi_x(\Sigma)$ of example 8, and the following relation R:

	A	B	C	D
u_1	1	3	5	7
u_2	2	3	6	8
u_3	2	4	5	9

Obviously, R obeys $AC \rightarrow D$ and $BC \rightarrow D$ and the curious dependency. Nevertheless, there does not exist an extension of R in SAT (Σ).
Using our algorithm we obtain

Ω_4	t_1	t_2	A	B	C	D	E
	u_1	u_2	0	1	0	0	1
	u_1	u_3	0	0	1	0	1
	u_2	u_3	1	0	0	0	1

and $\varsigma_C \wedge \varsigma_E \not\leq \varsigma_D$. This contradicts the dependency $CE \rightarrow D$.
Maier [1] defines the following additional condition: if $u_1[A] = u_3[A], u_2[B] = u_3[B]$, and $u_1[C] = u_2[C]$ then $u_1[D] = u_2[D]$. The relation R_1 used in examples 5, 10, 13 indicates that this condition is not sufficient. It can be shown that for any k there does not exist an equality formula

$$\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_k \rightarrow \alpha$$

which could be used as a necessary and sufficient condition for the extensibility of relations in SAT $(\pi_x(\Sigma))$ to relations in SAT (Σ).
Ginsburg and Zaiddan [15] have shown that the projection of Fd-families is not necessarily an FD-family. In example 14 it is possible to show that no Horn formula can be used to express conditions for the extensibility of relations in SAT $(\pi_x(\Sigma))$ to relations in SAT (Σ) (see [8]). For example 14 an equality formula similar to the 'curious' dependency presented in example 15 has the form

$$(\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \alpha_4) \rightarrow (\beta_1 \vee \beta_2 \vee \ldots \vee \beta_9).$$

Formulae of this form are clearly not Horn formulae.

5 Preserving Constraints

We would like to know the conditions under which whenever the relation $S(X)$ satisfies $\pi_X(T(\Sigma))$ it follows that $S(X)$ is a projection of a relation SAT $(\Sigma(U))$. The relations discussed in examples 14 and 15 show that if Σ is a set of functional dependencies over U then functional dependency families or classes are not preserved under projection.

It should be noted that the above mentioned problem is a special case of the database satisfaction problem. To solve this problem, some definitions are needed.

An all 1's row or subrow over columns X of T will be denoted by $< 1 >_X$. The X may be dropped when it is understood. Similarly, an all 0's subrow over X is denoted by $< 0 >_X$. If a row or subrow over X has at least one zero then it is denoted by $< *0 >_X$.

Let Z^n denotes the $2^n0 - 1$ strings of length n. Given the c-tableau $T(X)$ then $T_Z(U)$ denotes the c-tableau that is constructed as the Cartesian product

$T(X) \times Z^n$ where $n = |U - X|, X \subseteq U$. For example if $U = ABC, X = AB$, and $T(X)$ is the following c-tableau:

A	B
0	0
0	1

when $n = 1, Z^n = \{0, 1\}$; and $T_Z(U)$ is the following c-tableau:

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1

We define a new table called oc-tableau, denoted as $\overset{\circ}{T}(X)$, of a given c-tableau $T(X)$ as follows:

$$\overset{\circ}{T}(X) = T(X) \bigcup \{< 1 >_X\}$$

That is, $\overset{\circ}{T}(X)$ is constructed from $T(X)$ plus the all 1's row over X.

Definition: The c-join of $T_1(X_1)$ and $T_2(X_2)$, written $T_1 \bar{*} T_2$, is the c-tableau $T(X_1X_2) = \overset{\circ}{T}_1(X_1) * \overset{\circ}{T}_2(X_2)$.

The c-join operation is defined in terms of the join operation after adding the 1's rows to the c-tableaux participating in the c-join. Notice that $T(X_1X_2)$ in the definition above is a c-tableau and not an oc-tableau, thus the all 1's row is eliminated in $T(X_1X_2)$. By definition the c-tableau does not include an all 0's row.

Example 17. Let $T_1(AB) = \{(0, 1)\}$, and $T_2(BC) = \{(1, 0)\}$, then:

	A	B	C
	0	1	0
$T_1 * T_2 =$	0	0	1
	0	1	0

Let $T_1(AB) = \{(0, 1), (0, 0)\}$ and $T_2(BC) = \{(0, 1), (0, 0)\}$, then:

	A	B	C
	0	1	1
$T_1 * T_2 =$	0	0	1
	0	0	0
	1	0	0

Example 18. Let $\pi_{ABCD}(T)$ be the c-tableau shown in Example 3. As it is discussed in Example 1, $U = ABCDE$ and $U - ABCD = E$ hence, $Z = \{(0), (1)\}$. Figure 4 shows $T_Z(U)$. If $T = T_Z(U)$, we can claim that whenever a relation $R(X)$ satisfies $\overline{\pi}_X(T)$ it follows that $R(X)$ is a projection of a relation SAT (T).

Theorem 6 *SAT* $(\overline{\pi}_X(T(\Sigma))) = \pi_X(SAT(T(\Sigma)))$ *iff* $T(U) = T_Z(T)$.

A	B	C	D	E
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	1	1
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1
1	1	1	1	0

Figure 4 : T_Z of example 18

Proof: Suppose that SAT $\overline{\pi}_X(T(\Sigma))) = \pi_X(\text{SAT}(T(\Sigma)))$. Thus $S_1(X) \in \text{SAT}(\overline{\pi}_X(T(\Sigma)))$ may be joined with any relation $S_2(U-X)$ producing $R = S_1 * S_2$ where $R \in \text{SAT}(T(U))$ since $X \cap (U-X) = 0$. All possible relations over $U - X$ are projections of $R \in \text{SAT}(T)$. Thus, a subrow $t[U - X]$ in $T(U)$ is an element in $Z^n, n = |U - X|$; and $T(U) = T_Z(T(U))$, where $T_Z(T(U))$ is the Cartesian product $\overline{\pi}_X(T(U)) \times Z^n$. Suppose that $T_Z(T(U)) = T(U)$. If $S_1(X)$ satisfies $\overline{\pi}_X(T), X \subseteq U$, then for any relation $S_2(U - X)$, the join $S_1 * S_2$ satisfies $T(U)$. Conversely, if $R(U)$ satisfies $T(U)$ then any projection $\pi_X(R)$ satisfies $\overline{\pi}_X(T)$. Thus $SAT(\overline{\pi}_X(T(\Sigma))) = \pi_X SAT(T(\Sigma)))$.

6 Conclusion

Given a set of propositional dependencies $\Sigma(U)$ and a relation S over $X \subseteq U$, we have identified conditions under which there exists a relation $R \in \text{SAT}(\Sigma(U))$ such that $S = \pi_X(R)$. Also, we have identified the conditions under which whenever the projection $\pi_X(I), X \subseteq U$ satisfies π_X SAT $(T(\Sigma))$ it follows that I satisfies $T(\Sigma)$, where $I =$ SAT $(\Sigma(U))$.

These results are applicable to functional dependencies since they are special type of propositional dependencies. Only propositional dependencies are utilized whereas in [6], [8] and [10] non-two-tuple constraints are suggested. The theoretical significance of our results is clear since our approach is completely new. The practical significance of these results in the area of the relational database is similar to the previously mentioned work.

References

[1] D. Maier: The theory of relational databases. Computer Science Press, Englewood, 1983.

[2] C. Delobel, M. Adiba: Relational databases. North-Holland, Amsterdam, 1985.

[3] J. D. Ullman: Principles of database systems. Computer Science Press, Rockville, 1980.

[4] G. Vossen: Datenbankmodelle, Datenbanksprachen and Datenbank-Management-System. Addison-Wesley, Bonn, 1987.

[5] B. Thalheim: Open problems in Database Theory. Bulletin EATCS, 40, 1989.

[6] P. Atzeni, R. Torlone: Approaches to updates over weak instances. Lecture Notes in Computer Science 364, 1989, 12-23.

[7] B.E. Jacobs, A.R. Aronson, A.C. Klug: On interpretations of relational languages and solutions to the implied constraint problem. ACM TODS, 7,2, 1982, 291-315.

[8] S.S. Al-Fedaghi, B. Thalheim: Logical foundation for two-tuple constraints in the relational database model. Kuwait 1988. Submitted for publication.

[9] C. Delobel, R.G. Casey: Decomposition of data base and the theory of Boolean switching functions. IBM J. Res. Dev., 17, 5, 1973, 374-386.

[10] Y. Sagiv, C. Delobel, D.S. Parker, R. Fagin: An equivalence between relational database dependencies and a fragment of propositional logic. JACM, 28, 3, 435-453.

[11] S.S. Al-Fedaghi: Dependency theory. Draft copy of this book is available as Technical Report TR 85-0202, Electrical and Computer Engineering Department, Kuwait University, Kuwait 1985.

[12] B. Thalheim: Functional dependencies in relational databases. Journal Inf. Process and Cybernetics, 21,1/2, 1985, 23-33.

[13] J. Berman, W. J. Blok: Positive Boolean dependencies. University of Chicago, Research Report in Computer Sciences, No. 5, June 1985.

[14] R. Fagin: Horn clauses and database dependencies. JACM 29, 4, 1982, 952-985.

[15] S. Ginsburg, S. M. Zaiddan: Properties of functional dependency families. JACM 29,3, 1982, 678-698.

[16] E. Sciore: Improving database schemes by adding attributes. ACM PODS, Atlanta, Georgia, 1983, 379-382.

[17] J. Demetrovics, Gy. Gyepesi: On the functional dependency and some generalizations of it. Acta Cybernetica 5(1981), 295-305.

(Received February 12, 1990)

Subscription information:

For Albania, Bulgaria, China, Cuba, Czechoslovakia, German Democratic Republic, Korean People's Republic, Mongolia, Poland, Romania, USSR, Vietnam
orders should be addressed to:

Kultura
Hungarian Foreign Trading Co.
H-1389 Budapest 62
P.O. Box 149
Hungary

For all other countries orders should be adressed to:

J. C. Baltzer AG
Scientific Publishing Company
Wettsteinplatz 10
CH-4058 Basel
Switzerland

Mailing address for editorial correspondence:

Acta Cybernetica
Árpád tér 2.
Szeged
H-6720 Hungary

INDEX — TARTALOM

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1990. dec.
Terjedelem: 11,2 (A/5) ív
Készült ofszetnyomással
az MSZ 6601 és az MSZ 5602—55 szabvány szerint
90-7063 — Szegedi Nyomda — Felelős vezető: Kónya Antal mb. igazgató