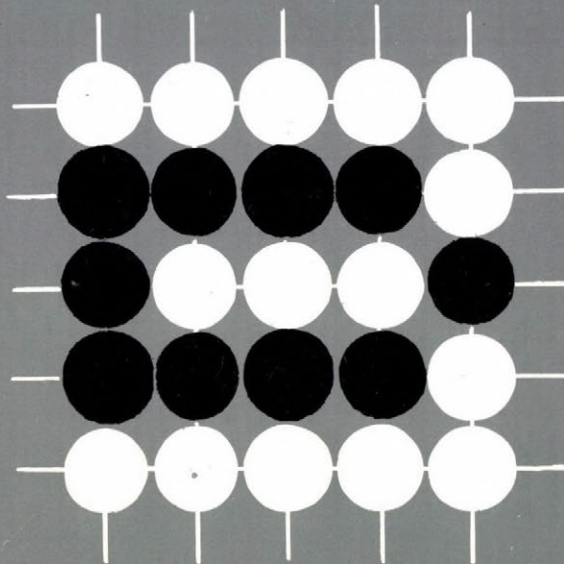




1986 JAN 27

TA Számítástechnikai és Automatizálási Kutató Intézet

Budapest





MAGYAR TUDOMÁNYOS AKADÉMIA  
SZÁMÍTÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

K Ö Z L E M É N Y E K

Szerkesztőbizottság:

DEMETROVICS JÁNOS (felelős szerkesztő)  
UHRIN BÉLA (titkár)  
GERTLER JÁNOS, KEVICZKY LÁSZLÓ,  
KNUTH ELŐD, KRÁMLI ANDRÁS, PRÉKOPA ANDRÁS

Felelős kiadó:

Dr. VAMOS TIBOR

ISBN 963 311 196 X

ISSN 0133-7459

C O N T E N T S

	page
H. Арато: О наблюдении авторегрессий высших порядков с непрерывным временем в дискретных точках .....	5
N. BUKOVSKI: An experience with dynamic data independence .....	13
J. DEMETROVICS, HO THUAN, NGUEN THANH THUY: Optimization of selections in relational expressions .....	27
J. DEMETROVICS, HO THUAN, NGUYEN XUAN HUY, LE VAN BAO: Balanced relation scheme and the problem of key representation .....	51
A. Гал: О сложности реализации одной треугольной матрицы вентильными схемами разной глубины .....	81
DANG VAN HUNG: Notes on trace languages, projection and synthesized computation systems ....	87
PHAM NGOC KHOI: A data structure for quadtree codes and application .....	105
HOANG KIEM, PHAM NGOC KHOI: An efficient synthesis of image matching algorithms .....	123
E. MUNIZ, M. FONFRIA, JORGE DE LA CANTERA: Experiences in the use of two database management systems for microcomputers ...	141
S. SHTRAKOV: On the c-separable and dominant sets of variables for the functions .....	155
VU DUC THI: Some special Sperner-systems .....	163



О НАБЛЮДЕНИИ АВТОРЕГРЕССИЙ ВЫСШИХ ПОРЯДКОВ С НЕПРЕРЫВНЫМ  
ВРЕМЕНЕМ В ДИСКРЕТНЫХ ТОЧКАХ

Арато Николай

Университет имени Етвеша Лоранда, Будапешт

Хорошо известно, что процесс авторегрессии первого порядка с непрерывным временем наблюдаемый в дискретных точках является также процессом авторегрессии (см [2]). Естественен вопрос: справедливо ли это для процессов авторегрессии высших порядков? Ответ в общем случае отрицателен, в дискретных точках мы получаем смешанный процесс авторегрессии-скользящего среднего. Я показываю это для процесса авторегрессии  $r$ -го порядка, а для процесса авторегрессии 2-го порядка считаю точные коэффициенты. В книгах [1], [2] не даются явные формулы для переписки в стохастические разностные уравнения дискретного времени.

Пусть  $\xi(t)$  процесс авторегрессии  $r$ -го порядка, т.е.,  $\xi(t)$  удовлетворяет дифференциальному уравнению:

$$(1) \quad d\xi^{(p-1)}(t) + a_1 \xi^{(p-1)}(t) + \dots + a_p \xi(t) dt = dw(t),$$

где  $w$  винеровский процесс (процесс брауновского движения), с коэффициентом диффузии  $\sigma^2$ , а многочлен

$$(2) \quad P(z) = \sum_{k=0}^p a_k z^{p-k} = \prod_{k=1}^p (z - \mu_k)$$

имеет корни с отрицательной реальной частью.

Нас интересует каким будет процесс  $x(n) = \xi(n\delta)$ , где  $\delta > 0$  заданное число.

Обозначив через  $\xi(t)$   $p$ -мерный процесс:  $\xi^*(t) = (\xi(t), \dots, \dots, \xi^{p-1}(t))$ , где  $*$  обозначает транспонирование.

Мы получаем:  $d\underline{\xi}(t) = A\underline{\xi}(t)dt + d\underline{w}(t)$ , где  $\underline{w}$   $p$ -мерный винеровский процесс с коэффициентом:

$$B_{\underline{w}} = \begin{pmatrix} & & & \\ & & & \\ & & \sigma^2 & \\ & & & \end{pmatrix}_{p \times p},$$

а

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & \\ 0 & 0 & 1 & 0 & \dots \\ & & & \ddots & \\ & & & & \ddots \\ -a_p & \dots & & & -a_1 \end{pmatrix}_{p \times p}.$$

Известно ([2]), что обозначив через  $\eta(n) = \xi(n\delta)$ :

$$(3) \quad \underline{\eta}(n) = Q\underline{\eta}(n-1) + \underline{\varepsilon}(n),$$

где  $Q = e^{A\delta}$ ,  $\underline{\varepsilon}(n)$  гауссовский белый шум.

Докажем следующую лемму:

Лемма: Пусть  $\underline{y}(n)$   $p$ -мерный процесс авторегрессии первого порядка:

$$\underline{y}(n) = S \underline{y}(n-1) + \underline{\varepsilon}(n),$$

тогда спектральная плотность  $r$ -го компонента  $\underline{y}(n)$ :

$$(4) \quad \frac{|T_r(e^{-i\lambda})|^2}{|\tilde{P}(e^{-i\lambda})|^2},$$

где  $\tilde{P}(z) = \sum_0^p \tilde{a}_n z^n$  характеристический многочлен матрицы  $S$ , а  $T_r$



многочлен  $(p-1)$  степени.

Доказательства: Спектральная плотность  $\underline{y}$  равняется (см. [1], 108. стр.):

$$f_{\underline{Y}}(\lambda) = \frac{1}{2\pi} (I - e^{-i\lambda} S)^{-1} B B^* (I - e^{i\lambda} S)^{-1},$$

где  $B B^* = E \underline{\varepsilon}(n) \underline{\varepsilon}^*(n)$ ,  $B_{\underline{Y}} = E \underline{y}(n) \underline{y}^*(n)$ ,  $B_{\underline{Y}} = S B_{\underline{Y}} S^* + B B^*$ .  
 $\underline{y}$  стационарен, поэтому  $\|S\| < 1$ , откуда:

$$(I - e^{-i\lambda} S)^{-1} = \sum_{k=0}^{\infty} e^{-i\lambda k} S^k.$$

Используя это и что  $\sum_{n=0}^p \tilde{a}_n S^n = 0$

$$\begin{aligned} \left( \sum_{n=0}^p a_n e^{i\lambda n} \right) (I - e^{-i\lambda} S)^{-1} &= \sum_{n=0}^p \tilde{a}_n e^{i\lambda n} \sum_{k=0}^{\infty} e^{-i\lambda k} S^k = \\ &= \sum_{n=0}^p a_n e^{i\lambda n} \sum_{k=0}^{n-1} e^{-i\lambda k} S^k + \left( \sum_{n=0}^p \tilde{a}_n S^n \right) \left( \sum_{\ell=0}^{\infty} e^{-i\lambda \ell} S^{\ell} \right) = \\ &= \sum_{k=0}^{p-1} e^{i\lambda k} R(k) e^{i\lambda}, \end{aligned}$$

где  $R(\ell) = R(\ell)_{p \times p}$  является матрицей размерности  $p \times p$ .

Из этого следует

$$f_{\underline{Y}}(\lambda) = \frac{1}{2\pi} \frac{1}{\left| \sum_{n=0}^p \tilde{a}_n e^{-i\lambda n} \right|^2} \left( \sum_{\ell=0}^{p-1} e^{i\lambda \ell} R(\ell) B \right) \left( \sum_{\ell=0}^{p-1} e^{-i\lambda \ell} R(\ell) B \right)^*$$

с элементом  $(r, r)$

$$(f_{\underline{Y}}(\lambda))_{rr} = \frac{|T_r(e^{-i\lambda})|^2}{|\tilde{P}(e^{-i\lambda})|^2},$$

$T_r$  многочлен  $(p-1)$ -ой степени, а это доказывает лемму.\*\*\*

В частном случае (1)-(3) характеристический многочлен  $Q = \prod_{k=1}^p (z - e^{\mu_k \delta})$ . Используя это и лемму сразу получаем:

Теорема: Наблюдая процесс авторегрессии  $p$ -го порядка с спектральной плотностью  $\frac{1}{\left| \prod_{k=1}^p (i\lambda - \mu_k) \right|^2}$  в точках  $n\delta$ ,

$(n=0, 1, \dots)$ , мы получаем смешанный процесс авторегрессии  $p$ -го порядка скользящего среднего  $(p-1)$ -го порядка с спектральной плотностью  $\frac{|T(e^{-i\lambda})|^2}{\left| \prod_{k=1}^p (e^{i\lambda} - e^{\mu_k \delta}) \right|^2}$ , где  $T$  многочлен  $(p-1)$ -ой степени.

Рассмотрим отдельно случай  $p=2$ .  $P$  имеет два корня:  $\mu_1$  и  $\mu_2$ . Из сказанной теоремы следует, что  $\xi(n\delta)$  удовлетворяет уравнению

$$\xi(n\delta) - (e^{\mu_2 \delta} + e^{\mu_1 \delta}) \xi((n-1)\delta) + e^{\mu_2 \delta + \mu_1 \delta} \xi((n-2)\delta) = t(n\delta),$$

где  $t(n\delta)$  процесс скользящего среднего 1-ого порядка. Я хочу подсчитать автоковариации этого процесса. Для легкости предположим  $\delta=1$  и обозначим  $\tilde{a} = -(e^{\mu_1} + e^{\mu_2})$ ,  $\tilde{a}_2 = e^{\mu_1 + \mu_2}$  и

$$t(n) = \xi(n) + \tilde{a}_1 \xi(n-1) + \tilde{a}_2 \xi(n-2),$$

тогда

$$E t^2(n) = (1 + \tilde{a}_1^2 + \tilde{a}_2^2) E \xi^2(n) + (2\tilde{a}_1 + 2\tilde{a}_1 \tilde{a}_2) E \xi(n) \xi(n-1) + 2\tilde{a}_2 E \xi(n) \xi(n-2)$$

$$E t(n)t(n+1) = (\tilde{a}_1 + \tilde{a}_1 \tilde{a}_2) E \xi^2(n) + (1 + \tilde{a}_2 + \tilde{a}_1^2 + \tilde{a}_2^2) E \xi(n) \xi(n-1) + (\tilde{a}_1 + \tilde{a}_1 \tilde{a}_2) E \xi(n) \xi(n-2) + \tilde{a}_2 E \xi(n) \xi(n-3).$$

Используем, что ([1], 118. стр.):

$$E \underline{\xi}(s+t) \underline{\xi}^*(s) = e^{At} B_{\xi} \quad B_{\xi} = (B_{ij})$$

где  $B_{\xi}$  удовлетворяет уравнению

$$A B_{\xi} + B_{\xi} A = -B_{\underline{w}}$$

откуда используя  $Q^2 + \tilde{a}_1 Q + \tilde{a}_2 I = 0$ , ( $Q = e^A$ ), получим

$$E t^2(n) = (1 + \tilde{a}_1^2 - \tilde{a}_2^2) B_{11} + 2\tilde{a}_1 (Q B_{\xi})_{11} ,$$

$$E t(n) t(n+1) = \tilde{a}_1 B_{11} + (1 + \tilde{a}_2)(Q B_{\xi})_{11} .$$

В случае  $\mu_1 \neq \mu_2$

$$A = \begin{pmatrix} -\mu_2 & 1 \\ -\mu_1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{pmatrix} \begin{pmatrix} -\mu_2 & 1 \\ -\mu_2 & 1 \end{pmatrix} = \frac{1}{\mu_1 - \mu_2} \begin{pmatrix} 1 & -1 \\ \mu_2 & -\mu_2 \end{pmatrix} \begin{pmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{pmatrix} \begin{pmatrix} -\mu_2 & 1 \\ -\mu_1 & 1 \end{pmatrix}$$

откуда выходит

$$Q = \frac{1}{\mu_1 - \mu_2} \begin{pmatrix} \mu_1 e^{\mu_2} - \mu_2 e^{\mu_1} & e^{\mu_1} - e^{\mu_2} \\ \mu_1 \mu_2 (e^{\mu_2} - e^{\mu_1}) & \mu_1 e^{\mu_1} - \mu_2 e^{\mu_2} \end{pmatrix} ,$$

и

$$B_{\xi} = \frac{\sigma^2}{-2(\mu_1 + \mu_2)\mu_1\mu_2} \begin{pmatrix} 1 & 0 \\ 0 & \mu_1\mu_2 \end{pmatrix} .$$

В случае  $\mu_1 = \mu_2 = \mu$  результат:

$$Q = e^{\mu} \begin{pmatrix} 1 - \mu & 1 \\ \mu^2 & \mu + 1 \end{pmatrix} ,$$

$$B_{\xi} = \frac{\sigma^2}{-4\mu^3} \begin{pmatrix} 1 & 0 \\ 0 & \mu^2 \end{pmatrix} .$$

Проведя подсчеты получаем в случае  $\mu_1 \neq \mu_2$

$$E t^2(n) = \frac{\sigma^2}{-2(\mu_1 + \mu_2)\mu_1\mu_2(\mu_1 - \mu_2)} \left[ (\mu_1 - \mu_2)(1 - e^{2(\mu_1 + \mu_2)}) + \right. \\ \left. + (\mu_1 + \mu_2)(e^{2\mu_1} - e^{2\mu_2}) \right],$$

$$E t(n)t(n+1) = \frac{\sigma^2}{-2(\mu_1 + \mu_2)\mu_1\mu_2(\mu_1 - \mu_2)} \left[ -\mu_1 e^{\mu_1} + \mu_2 e^{\mu_2} + e^{\mu_1 + \mu_2} \right. \\ \left. (\mu_1 e^{\mu_2} - \mu_2 e^{\mu_1}) \right],$$

а в случае  $\mu_1 = \mu_2 = \mu$

$$E t^2(n) = \frac{\sigma^2}{-4\mu^3} \left[ 1 + 2e^{2\mu}(1 + \mu) - e^{4\mu} \right],$$

$$E t(n)t(n+1) = \frac{\sigma^2}{-4\mu^3} \left[ -e^{\mu}(1 + \mu) + e^{3\mu}(1 - \mu) \right].$$

Замечание 1. Важность вычисления коэффициентов смешанного процесса авторегрессии скользящего среднего объясняется тем, что мы не можем наблюдать производные процесса, если бы это было возможно, мы могли бы использовать формулу (3).

Замечание 2. Можно было бы попробовать считать спектральную плотность дискретного процесса зная форму спектральной плотности процесса с непрерывным временем. Но этот путь видимо значительно сложнее (кроме 1-ого порядка) поскольку, если  $f(\lambda)$  спектральная плотность процесса с непрерывным временем, то спектральная плотность процесса наблюдаемого в дискретных целых точках дается формулой

$$\hat{f}(\lambda) = \sum_{k=-\infty}^{\infty} f(\lambda - (2k+1)\pi).$$

Замечание 3. Показав, что  $\sum_{j=1}^k v_j(n)$  скользящее среднее, где  $v_j(n) = \sum_{i=0}^p a_{ji} u_j(n-i)$  скользящее среднее  $p$ -го порядка, для которых:  $E u_j(n) u_\ell(n-m) = 0$  ( $j \neq \ell$ ,  $m > 0$ ), можно доказать лемму, не используя спектральную плотность. Для незасисимых скользящих средних это показано в [3]. Похожим методом доказывается случай  $k=2$ . Правильен ли результат для  $k \geq 3$ , мне неизвестно.

#### Литература

- [1] Arató, M. (1982), Linear stochastic systems with constant coefficients. A statistical approach. Lecture Notes in Control and Information Sciences, 45, Springer Verlag, Berlin, Heidelberg.
- [2] Липцер, Р. Ш., Ширяев, А. Н. (1974), Статистика случайных процессов, Наука, Москва.
- [3] Granger, C. W. J. (1972), Time series modelling and interpretation, European Econometric Congress, Budapest.

Folytonos idejű magasabbrendű autoregressziók megfigyelése  
diszkrét pontokban

ARATÓ MIKLÓS

Összefoglaló

Folytonos idejű  $p$ -adrendű autoregresszióról megmutatom, hogy  $ARMA(p, p-1)$  lesz diszkrét helyeken nézve. Általános esetben kiszámolom spektrálsűrűségének nevezőjét. Ehhez megmutatom, hogy néz ki egy elsőrendű többdimenziós autoregresszió egy komponensének spektrálsűrűsége (4). A  $p=2$  esetben kiszámolom a folyamat mozgóátlag részének autokovarianciáit, és ezzel az  $ARMA(2,1)$  folyamat együtthatóit (v.ö. [2], 257 oldalon szereplő közelítésekkel).

Continuous time autoregression processes with discrete time  
observations

ARATÓ Nicolay

Summary

In this paper we prove that the  $p$  order autoregressive process  $\xi(t)$  with continuous time parameter,  $AR(p)$ , becomes a discrete time  $ARMA(p, p-1)$  one. Formula (4) gives the general form of spectral density of the  $r$ -th component of a discrete time first order  $p$ -dimensional autoregressive process  $\underline{y}(t)$ . In the special case  $p=2$  all the calculations are carried out, i.e., the coefficients of  $ARMA(2,1)$  are calculated (see approximations in [2], p. 257).

## AN EXPERIENCE WITH DYNAMIC DATA INDEPENDENCE

*N. BUKOVSKI*Institute "Interprograma"  
Sofia, Bulgaria

## I. INTRODUCTION

Information handling in Data Dictionaries (DD) is maintained by program components, performing the data access, the data storing and organization. One way to implement such program components is by using Data Base Management Systems (DBMS): self-made or packages. In this paper we will discuss how a DBMS, as a part of the DD, can be used to satisfy the DD requirements for handling its information. For simplicity purposes by DBMS we will imply the program components, realizing exactly this DD function (although it would be more precise to denote it as DD DBMS). The DD has a fixed logical data structure, but nevertheless some changes could be made during the processes of DD installation and operation, e.g., adding user's information types, or physical data restructuring. This requires from the DBMS, upon which the DD is built, to support dynamic data independence. So the rest of the DD software is isolated, thus avoiding its recoding or recompilation. The way the dynamic data independence concept was incorporated into the DBMS of a Data Dictionary is presented in this paper. The DD concerned is an integrating tool in the PLUS complex - a program development environment [1], [2] and provides information for the numerous PLUS components. It, together with the DD extensibility feature, supporting user-defined information types, requires that all programs, accessing the DD, should be independent of its data structure and organization. We shall discuss how a self-made (autonomous) DBMS was designed to support dynamic data independence together with the advantages and drawbacks of the approach chosen.

## II. INITIAL REQUIREMENTS

A DD consists of two software components: a DBMS, accessing the DD database, and functional software, implementing the DD functions - selective report generation, cross-referencing, etc. The information contained in the DD database comprises entities of different classes and the relationships among them. Each entity or relationship type consists of attributes: description, keywords, etc.

Considering the DD information and functions needed and its including in the PLUS complex, the following initial requirements for the DD were established:

(1) Data independence. The DBMS should allow changes in the data structure without recoding or recompilation of the application programs accessing them (by application program we imply here and later in the paper any program of the DD functional software or of the PLUS environment which accesses the DD information). These changes include:

- logical structure modifications, for example, the adding of entity attributes or classes;
- physical data restructuring intended to improve DBMS performance in compliance with the specific conditions of the DD usage. For example, another segmentation of the logical record can be chosen, depending on the specific usage frequency of its data elements, or the parameters of the hash address method can be adjusted to the prevalent data volume or changeability.

We will point out that these changes are required in the process of the DD operation which affects especially strongly the PLUS components accessing the DBMS of the DD.



- (2) Interface simplicity. The simplicity of the interface, i.e. of the data manipulation statements, used to interface with the DBMS, requires that application program should not be concerned with the variety and complexity of the DD information. This has two aspects. The first one requires that application programs, accessing the DD database, should know only the data they are processing, not the DD information as a whole. The second objective requires a unified access method to be provided, irrespective of the logical type of the data needed: records or relationships. They should be processed in a uniform way, requiring a key - simple or composed, and a list of the necessary attributes. This requirement is influenced by the simplicity of the relational data model, dealing only with relations ("flat" files) and avoiding data structuring concepts, e.g., linked records.

### III. PRELIMINARY DESIGN DECISION ON DATA INDEPENDENCE

We will make two preliminary decisions, allowing fulfillment of the first objective: data independence.

- (1) Data independence requires the DBMS to perform transformation (mapping) of the data according to their schemes: external, logical and internal [3]. A choice has to be made about the moment data mapping is performed: at run time (dynamic), or before calling the DBMS (static). In [3] dynamic data independence is recommended when supporting unplanned (ad hoc) queries. The DD supports exactly such queries, providing reporting and interrogation facilities. The criteria, through which the DD data are selected, are so numerous, that preliminary planning of all queries and their structures is impossible, especially with the DD extensibility feature. The solution to derive the

queries data structure from the logical schema of the DD information burdens the programs processing these queries with the complexity of the whole logical scheme. So the only solution is to allow the definition of the data needed, i.e. the program view of the data it processes, to be done at run time.

- (2) Another aspect of data independence concerns the PLUS components, which interface with the DBMS. Any change in the data structure or organization must not affect their programs. They have to be totally insulated, even their recompilation is not acceptable (though this could be permitted for the DD programs). So this requires that modification of the logical and physical schemas should be allowed at any time without revision of the programs.

These two considerations require that the DBMS should support dynamic data independence, both at logical and physical level.

#### IV. INTERFACE DEFINITION

The first step of the DBMS design will be defining of its data manipulation statements. The DBMS will be treated as a "black box", which provides the interface to the DD information. The second step will be designing of this black box, i.e. the DBMS.

The dynamic data independence we specified requires that programs, interfacing with the DBMS, should define in the data manipulation statements the structure of the data they process. On the other hand this definition has to be represented in a table-oriented way (as relations), thus satisfying the objective for interface simplicity. So the first step in the process of the interface definition will be specifying these relations.

There are two types of relations, corresponding to the DD information:

- entity relation. Each entity relation represents an entity class: every tuple of the relation corresponds to an entity, every domain - to an attribute. So a separate relation is maintained by the DBMS for each entity class. The entity relation is represented as

$$R(ID, A_1, A_2, \dots)$$

where  $R$  - the type of the relation, i.e. the entity class;

$ID$  - the relation key, identifying its tuples (usually it is the entity name);

$A_1, A_2, \dots$  - the list of the entity attributes, composing this relation:

- relationship relation. A separate binary relation is maintained for each couple of entity classes, relationship between which is allowed. Each relation tuple corresponds to a relationship between two entities, every relation domain - to an attribute, describing this relationship (the so called intersection data). This type of relation is represented as

$$R(ID_1 \star ID_2, A_1, \dots)$$

where  $R$  - the relation type;

$ID_1 + ID_2$  - the composed key of this relation, identifying its tuples.  $ID_1, ID_2$  are the keys of the two entity classes;

$A_1, A_2, \dots$  - the list of the relationship attributes (intersection data).

Now we shall define two data manipulation statements, using some notions of the relational data languages [4], [5]. We will point out that before using these operations, the DBMS user is provided with all relations, maintained by the DBMS and describing its information. For simplicity, only one type of data access will be described - data retrieval.

The two datamanipulation operators are as follows:

- FOR. This statement retrieves sequentially the tuples of a given entity relation. Each time the statement is performed, a tuple (i.e. an entity) is derived and only the attributes, specified in the statement, are moved. The statement has the following format:

*FOR R(A<sub>1</sub>, A<sub>2</sub>, ...)*

- PREDICATE. This statement sequentially processes only these entities or relationships of relations, which satisfy the condition specified in the statement. Each time the statement is performed, a tuple (i.e. an entity or a relationship) satisfying the condition is retrieved. When given an entity relation, the condition contains the name of the entity, which is to be retrieved. When given a relationship relation, the condition contains one or two names (keys) and all tuples, containing them in their composed key, are retrieved. If one entity name is specified, then all relationships of this entity are traced; if two entity names, then the relationship between these entities is processed. The statement has the following format:

*PREDICATE R(A<sub>1</sub>, A<sub>2</sub>, ...): ID<sub>1</sub> = X [, ID<sub>2</sub> = Y]*

where ID<sub>1</sub>, ID<sub>2</sub> - specify the condition;

X, Y - names, used in the condition.

For example, if having a relationship "R<sub>12</sub>" between the entity classes "C<sub>1</sub>" and "C<sub>2</sub>" with relationship attributes "A<sub>1</sub>, A<sub>2</sub>, ..., A<sub>11</sub>", the relationship between two entities with keys "NAME<sub>1</sub>" and "NAME<sub>2</sub>" can be traced using the following statement:

*PREDICATE R<sub>12</sub> (A<sub>1</sub>, A<sub>3</sub>, A<sub>4</sub>): ID<sub>1</sub> = NAME<sub>1</sub>, ID = NAME<sub>2</sub>*

and as a result the relationship attributes "A<sub>1</sub>, A<sub>3</sub>, A<sub>4</sub>" will be retrieved.

Finally, two features of the two statement will be mentioned:

- by means of the attribute list the programs are able to specify only the attributes they need, not all of the relation attributes;
- the list of the attributes can comprise only attributes, belonging to the corresponding relation. This restriction means that the list can be only a subset of the relation.

## V, DD DBMS DESIGN

The main feature of the DBMS discussed is the dynamic data independence, so the DBMS design will be presented having in mind primarily this aspect.

Dynamic data independence means that a run time the DBMS performs a mapping between the three schemas: external, logical and integral. This requires that the three schemas should be *interpreted* at run time, when calling the DBMS. In this way they can be modified at any time without amending the programs using them. On the other hand interpretation of the schemas requires their storing in object format, as *coded tables*.

We already discussed the way the programs provide their view of data (i.e. the external schema) in the statements FOR and PREDICATE. This allows an easy transformation of these operators into coded tables to be done at compilation or at run time. Now we will discuss the way the logical and physical schemas are formed as coded tables in order to allow their interpretation.

The logical schema, describing the logical structure of the DD information, is built upon three types of tables:

- (1) Entity Structure Table (EST). A separate table is maintained for each entity class. This table describes

its contents of attributes, their logical sequence, and the key used to identify the entities.

- (2) Relationship Table (RT). A table is provided for every couple of entity classes a relationship between which is allowed. This table contains the keys of the classes (usually the entity names) and the contents and the logical sequence of the relationship attributes (intersection data).

These tables correspond to the relations, describing the program view of data (external schema) in the statements FOR and PREDICATE and allow specifying of the programmer's information needs in the data manipulation statements.

- (3) Attributes Table (AT). This table describes each entity or relationship attribute: format-variable or fixed, length, type-numerical or coded, etc.

The internal schema is presented by means of the Physical Organization Table (POT). It describes the physical data structure and organization: storage allocation, block and record length, addressing parameters, etc.

The DBMS architecture and the algorithm of logical and physical mapping are shown in Fig. 1. The sequence of the algorithm actions, represented as circle numbers, is the following:

- (1) The Application program calls the DBMS, providing the following information in the data manipulation statements:

- the type of the statement: FOR or PREDICATE;
- the type of the relation. We will note that from program point of view this type specifies an entity class or a relationship between two classes;

- the contents and the sequence of the attributes needed;
  - the condition for the PREDICATE operator;
  - the work area, in which the data are to be moved.
- (2) The Logical Mapping Processor reads the table, corresponding to the specified relation. From this table it derives the required attributes within the logical sequence of the attributes, composing the table.
- (3) The Logical Mapping Processor obtains from the attributes table the format and the length (if varied) of each attribute, specified by the application program.
- (4) The Logical Mapping Processor calls the Physical Mapping Processor, providing it with the information thus obtained.
- (5) The Physical Mapping Processor derives from the POT information about the physical location of the block, containing the required data, say volume, file relative block address, etc.
- (6) The Physical Mapping Processor calls the I/O Processor, providing it with the physical address of the necessary block.
- (7) The I/O Processor reads the block from the metadatabase into the buffer pool.
- (8) The Physical Mapping Processor, using the information about the attributes format and length and the records blocking, performs the following:
- locates the position of the required attributes within the block;
  - moves them into the work area of the application program in accordance with the sequence specified by it.

## VI. RESULTS

The so designed DBMS allows dynamic performance of the following changes, concerning the data structure and organization, without revision of the application programs which relied on the previous structure:

(1) Changes of the logical structure. They include changes in the following:

- entity contents and structure. For example, adding of attributes or changing their order are allowed. This affects only the corresponding entity structure table;
- relationship between entities (e.g., the relationship between two classes can be deleted). This results in the modification of the corresponding relationship table;
- attribute format, e.g., changing of the attribute length or type (fixed or variable), etc.

The logical dynamic independence permits the insulation of the programs, accessing the DBMS, from changing information requirements (these changes can be transparent at the programming level). This flexibility greatly simplifies the program maintenance.

(2) Physical organization modification. This includes changes in addressing parameters, record and block length, storage allocation, etc. These changes allow tuning in accordance with the DBMS usage in order to improve its performance.

Dynamic data independence thus achieved provides to additional advantages:

- flexibility. The application programs access the DBMS in a way, independent of the data structure and organi-



zation, so their algorithm is not tied to the particular data representation. This reduces the efforts needed when changing the programs;

- simplicity. The interface to the DBMS represents the data in a table-oriented way, using some ideas of the relational data model. This simplifies application program development and maintenance.

Generally, dynamic data independence maintenance has two drawbacks: first, maintenance of tables with data description, and second, run time overhead due to the interpretation of these tables. In our case, the choice of dynamic data independence is justified. The first disadvantage can be compensated since the DD functional software needs such tables in order to process the DD information in a unified way, regardless of its type. The second drawback - the overhead, is not so significant because the volume of the DD information is not too great.

## VII. CONCLUSION

This paper discusses the way a DBMS of a Data Dictionary was designed to support dynamic data independence. The experience is gathered during the design and implementation of the PLUS program development environment (though integrated with the PLUS complex, the DD can be used independently). The dynamic data independence, embedded in the DBMS of the Data Dictionary, greatly reduces the efforts for integrating the DD with the PLUS components. The experience and the results achieved can be used in all Data Dictionaries having an extensibility feature or integrated with a program development environment. Generally, the approach is applicable not only to DBMS of Data Dictionaries, but also to any DBMS with dynamic data independence as a key requirement.

## REFERENCES

1. Shoshlekov, I., PLUS language in the data base environment, Proceedings of the sixth international seminar on Data Base Management Systems, Hungary, 1983.
2. Bukowski, N., Metadata Handling in the PLUS Complex, Proceedings of the sixth international seminar on Data Base Management Systems, Hungary, 1983.
3. Martin, J., Computer Data Base Organization, Prentice-Hall, Inc., New Jersey, 1975.
4. Schmidh, J.W., Some high level language constructs for data of type relations, ACM Transactions on Data Base Systems, Vol. 2, No. 3, September 1977.
5. Beck, L.L., A Generalized Implementation Method for Relational Data Sublanguages, IEEE Transactions of Software Engineering, Vol. SE-6, No. 2, March 1980.

Dinamikus adat-függetlenséggel kapcsolatos tapasztalatok

N. BUKOVSKI

Összefoglaló

A cikkben a szerző bemutatja, hogyan lehet a dinamikus adat-függetlenség fogalmát felhasználni az Adat Szótárakkal kapcsolatos Adatbázis Kezelő Rendszerekben.

Опыты касающиеся динамической независимости данных.

Н. Буковски

Резюме

Показывается возможность использования понятия независимости данных для систем обработки баз данных связанных с словарями данных.



## OPTIMIZATION OF SELECTIONS IN RELATIONAL EXPRESSIONS

*J. DEMETROVICS, HO THUAN*

Computer and Automation Institute,  
 Hungarian Academy of Sciences  
 Budapest, Hungary

*NGUEN THANH THUY*

Hanoi Polytechnical Institute  
 Vietnam

ABSTRACT

One operation met usually in the relational expressions is the selection of a relation  $R$  on a conditional expression  $E = \bigwedge_{i=1}^n E_i$ . In this paper, basing upon the estimations of probability of the tuples  $r \in R$  satisfying  $E_i$ , we shall show one simple  $O(n \log n)$  algorithm, where  $n$  is the length of  $E$ , rearranging the sub-expressions of  $E$  and so, the average probabilistic complexity of the algorithm for finding

$$\sigma_E(R) = \{r \in R / r \text{ satisfies } E\}$$

is minimal.

§0. INTRODUCTION

On operation met usually in expressions of relational algebra is the selection of a relation  $\mathcal{R}$  on a conditional expression  $\mathcal{E}$ . In general, it requires time  $O(N)$ , where  $N$  is the number of the tuples in  $\mathcal{R}$ , to perform that selection. However, when it is

discussed in the common situation with respect to other operations, for instance, projection, join, cartesian product ... the following principle is in priority: "Perform the selections and the projections as early as possible."

The transformation

$$\sigma_{\mathcal{C}}(R) \Rightarrow \sigma_{\mathcal{C}_1}(\sigma_{\mathcal{C}_2}(\dots\sigma_{\mathcal{C}_m}(R)\dots))$$

when  $\mathcal{C}$  is of the form

$$\mathcal{C} = \bigwedge_{i=1}^m \mathcal{C}_i,$$

is performed for the above principle. When an initial parse tree of a relational expression is reduced to a better form by the general optimization principles for relational expressions [1,3,4], it is possible that in the obtained parse tree there is a conjunctive selection

$$\sigma_{E_1} \wedge \dots \wedge \sigma_{E_n}^{(R)}$$

of a relation R on

$$E = E_1 \wedge \dots \wedge E_n.$$

Due to the commutativity and the associativity of the operation  $\wedge$ , the relational expression

$$\sigma_{\bigwedge_{i=1}^n E_i}^{(R)}$$

can be reduced to

$$\sigma_{\bigwedge_{i=1}^n E_{\tau_i}}^{(R)}$$

where  $\tau = \{\tau_1, \dots, \tau_n\}$  is a permutation of  $\{1, \dots, n\}$ . That is why we want to find the best permutation  $\tau = \{\tau_1, \dots, \tau_n\}$  such that the time complexity (cost) to find  $\sigma_E(R)$  is minimal. In this paper, basing upon the estimations of probability of the tuples  $r \in R$  satisfying the logical expressions  $E_i$  and the

definition of the average probabilistic complexity of an algorithm, the best ordering  $\tau = \{\tau_1, \dots, \tau_n\}$  of the subexpressions  $E_i$ ,  $i = \overline{1, n}$  will be obtained such that the average probabilistic cost (complexity) of the algorithm finding  $\sigma_E(R) = \{r \in R / r \text{ satisfies } E\}$  is minimal.

When  $E$  is an arbitrary logical expression (as defined in §1) it can be reduced to the conjunctive - disjunctive normal form

$$E = \bigvee_{i=1}^n \bigwedge_{j=1}^{n_i} E_j^i$$

where  $E_j^i$  is of the form either  $A \theta B$  or  $A \theta c$  or  $c \theta A$ , where  $A, B$  are attributes,  $c$  is a constant and  $\theta$  is a comparison operator  $\theta \in \{\geq, >, <, \leq, =, \neq\}$ .

As the algorithm for a conjunctive selection can be used for a disjunctive selection with some modifications, so for a given arbitrary logical expression  $E$ , it is possible to find the best ordering of the subexpressions of  $E$  such that the average probabilistic cost of the algorithm finding  $\sigma_E(R)$  is minimal.

It is interesting that when the cost to find the best ordering is added to the cost of the algorithm finding

$$\sigma_E(R) = \sigma_{\bigwedge_{i=1}^n E_{\tau_i}}(R),$$

the total cost remains desirable i.e. is less than the cost of the algorithm finding

$$\sigma_E(R) = \sigma_{\bigwedge_{i=1}^n E_i}(R)$$

with large  $N$ . The algorithm shown here for finding the best ordering  $\tau$  of the subexpressions of  $E$  can be implemented in the computers as a subroutine without any access to the secondary memory devices containing the file  $R$ . Its time complexity is of  $O(n \log n)$  where  $n$  is the length of  $E$ .

§1. BASIC DEFINITIONS

Definition 1: A relation R with a set of attributes  $U = \alpha(R) = \{A_1, \dots, A_k\}$  and the corresponding ranges  $D_1, \dots, D_k$  is defined as follows

$$R \stackrel{\text{def}}{=} \{r: U \rightarrow \prod_{i=1}^k D_i \mid \forall i \ 1 \leq i \leq k \ r(A_i) \in D_i\}$$

or

$$R \stackrel{\text{def}}{=} \{r = \langle t_1, t_2, \dots, t_k \rangle \mid \forall i \ 1 \leq i \leq k \ t_i \in D_i\}$$

Each  $r \in R$  is called a tuple of R.

Definition 2: A logical expression E in R with the set of attributes  $U = \alpha(R)$  and the ranges  $D_1, \dots, D_k$  can be defined recursively as follows:

1) An expression of the form  $A \theta B$ ,  $A \theta c$ ,  $c \theta A$ ,  $A, B \in U$ ,  $c \in \prod_{i=1}^k D_i$ ,  $\theta \in \{\geq, >, \leq, <, =, \neq\}$ , is a simple logical expression.

2) If  $E_1, E_2$  are logical expressions, then

$$E_1 \vee E_2, E_1 \wedge E_2, \neg E_1 \text{ are also logical expressions.}$$

Definition 3: A logical expression E in R which is of the form  $E = E_1 \wedge \dots \wedge E_n$  is called conjunctive logical expression.

Definition 4: Given a logical expression E in a relation R with the set of attributes  $\nu$  and the ranges  $D_i, i = \overline{1, k}$ , and a tuple r of R.

We say that r satisfies E if when substituting the names of the attributes A in E by the value  $r.A_i \in \prod_{i=1}^k D_i$  of the tuple  $r \in R$ ,



the obtained logical expression has the value "true".

Definition 5: Given a relation R and a logical expression E. The selection of the relation R on the condition E, denoted by  $\sigma_E(R)$  is defined as follows:

$$\sigma_E(R) = \{r \in R \mid r \text{ satisfies } E\} .$$

If E is a conjunctive logical expression, the selection  $\sigma_E(R)$  is called a conjunctive selection.

Definition 6: Let  $\Omega$  be a probability space of finite cardinality, i.e. in  $\Omega$  is defined a probability measure  $\rho: 2^\Omega \rightarrow [0,1]$  satisfying the probability axioms. Put

$$\Omega = \{\omega_1, \omega_2, \dots, \omega_g\}$$

and

$$\rho_i = \rho(\{\omega_i\}), \quad i = \overline{1, g} .$$

Then, the average probabilistic value of the real valued function f defined on  $\Omega$  corresponding to the probability measure  $\rho$  is defined as

$$\bar{f}_\rho = \sum_{i=1}^g f(\omega_i) \rho_i .$$

## §2. AN APPROACH TO THE PROBABILITY ESTIMATIONS

Let a relation R be given with the set of attributes u and the ranges  $D_i, i = \overline{1, k}$ . As defined in definition 2, a logical expression can be constructed by the simple logical expressions and the logical operators  $\{\wedge, \vee, \neg\}$ .

The following statistical parameters obtained and collected during the manipulation of R can be estimated:

- 1) The distribution of values of the attributes in  $u$ .
- 2) The exact upper bound and lower bound for the attributes in  $u$  during the manipulation.
- 3) The number of distinct values of an attribute. One of the simple assumptions is that the values of every attribute  $X \in u$  are uniformly distributed in the segment  $[X_m, X_M]$  where  $X_m = \min R[X]$ ,  $X_M = \max R[X]$ . (H1).

With the assumption (H1), it is easy to give the probability estimations  $\Pr(E)$  for the tuple  $r \in R$  satisfying  $E$ .

For instance, for  $X, Y \in u$ ;  $a, b \in R[X]$ , we have

$$\Pr(X=a) = \frac{1}{\text{card } R[X]} ; \quad (2.1)$$

$$\Pr(X \geq a) = \begin{cases} \frac{X_M - a}{X_M - X_m} , & X_m \leq a < X_M \\ \frac{1}{\text{card } R[X]} , & a = X_M \end{cases} \quad (2.2)$$

$$\Pr(X > a) = \Pr(X \geq a) - \frac{1}{\text{card } R[X]} \quad (2.3)$$

$$\Pr(a \leq X < b) = \begin{cases} \frac{b-a}{X_M - X_m} , & X_m \leq a < b < X_M \\ \frac{X_M - a}{X_M - X_m} - \frac{1}{\text{card } R[X]} , & X_m \leq a < b = X_M \end{cases} \quad (2.4)$$

$$\Pr(X=Y) = 0 \quad \begin{array}{l} \text{if } X_m < X_M < Y_m < Y_M \\ \text{or } Y_m < Y_M < X_m < X_M \end{array} \quad (2.5)$$

$$\Pr(X=Y) = \frac{1}{\text{card } R[X] \cdot \text{card } R[Y]} \quad \begin{array}{l} \text{if } X_m < X_M = Y_m < Y_M \\ \text{or } Y_m < Y_M = X_m < X_M \end{array} \quad (2.6)$$

$$\Pr(X=Y) = \frac{1}{\max \left( \frac{X_M - Y_m}{X_M - X_m} \text{ card } R[X], \frac{X_M - Y_m}{Y_M - Y_m} \text{ card } R[Y] \right)} \quad (2.7)$$

$$\text{if } X_m < Y_m < X_M < Y_M$$

$$\text{or } Y_m < X_m < Y_M < X_M$$

$$\Pr(X=Y) = \frac{1}{\max(\text{card } R[X], \text{card } R[Y])} \quad (2.8)$$

$$\text{if } X_m = Y_m < X_M = Y_M$$

(a special case of (2.7)).

Remark 1. To compute  $\Pr(E)$  for a non simple logical expression, we use the following rules:

- a)  $\Pr(E_1 \wedge E_2) = \Pr(E_1) \cdot \Pr(E_2)$  where  $E_1, E_2$  are independent.
- b)  $\Pr(E_1 \vee E_2) = \Pr(E_1) + \Pr(E_2) - \Pr(E_1 \wedge E_2)$ ;
- c)  $\Pr(\neg E_1) = 1 - \Pr(E_1)$ .

Remark 2. In [2], for the computation of  $\Pr(X=Y)$ , the authors gave only formula (2.8), with the assumption (H1). Of course, when taking attention to the relative positions of the segments  $[X_m, X_M]$  and  $[Y_m, Y_M]$ , this simple formula is not complete.

Remark 3. For other distribution types, the idea of this paper is also useful. One must only considered an appropriate method for computing the probability estimations.

§3. THE MAIN PROBLEM

Given a relation R with the set of attributes  $U=\alpha(R)$  and the ranges  $D_i, i=\overline{1,k}$ . E is a logical expression in R. The selection  $T=\sigma_E(R)$  can be obtained by the following algorithm:

$$\left. \begin{array}{l}
 T:=\emptyset \\
 \text{for each } r \in R \text{ do} \\
 \quad \text{if test } (r,E) \text{ then add } r \text{ to } T
 \end{array} \right\} \quad (1)$$

The function test (r,E) performs two operations:

- i) Replaces the names of attributes  $A \in U$  by the values  $r.A$  of the tuple  $r \in R$ .
- ii) Computes the obtained logical expression and assigns the result to the function test.

Given a tuple  $r \in R$ . Denote  $\text{cost}(r,E)$  the cost paid to perform the function test (r,E). In this section, we always consider that E is a conjunctive logical expression

$$E = \bigwedge_{i=1}^n E_i .$$

The function test (r,E) can be computed by two different ways:

Way 1: Compute all functions test (r, $E_i$ ) and then set

$$\text{test}(r,E) = \bigwedge_{i=1}^n \text{test}(r,E_i).$$

We have the algorithm:

$$\begin{array}{l}
 \text{test}(r,E) := \text{true}; \\
 \text{for } i := 1 \text{ to } n \text{ do test}(r,E) = \text{test}(r,E) \wedge \\
 \quad \wedge \text{test}(r,E_i).
 \end{array} \quad (2)$$

The one-pass compilers compute often the conjunctive logical expressions in this way, for instance the compiler on the

hypothetical computer P-code for language PASCAL-S.

Way 2: It is obvious that if there is some  $i_0$  during the computation of test  $(r, E_{i_0})$  such that  $\text{test}(r, E_{i_0}) = \underline{\text{false}}$  then it is possible to conclude immediately that  $\text{test}(r, E) = \underline{\text{false}}$  and to halt the calculation of test  $(r, E)$ . This is expressed in the algorithm as follows.

Test $(r, E) := \underline{\text{false}}$	}	(3)
<u>for</u> $i := 1$ <u>to</u> $n$ <u>do</u>		
<u>if</u> $\neg \text{test}(r, E_i)$ <u>then</u> <u>goto</u> L;		
$\text{test}(r, E) := \underline{\text{true}} ;$		

L:

Intuitively, it is easy to see that the method in the algorithm (3) is very natural. (Of course it is better than the algorithm (2).) However, it is very interesting if we know the probabilities of the tuples  $r \in R$  satisfying the expressions  $E_i$  in  $R$  and so we can expect that there exist a best ordering of the subexpressions  $E_i$  such that the average probabilistic cost of the algorithm (3) is minimal.

To make clear this idea we do as follows:

At first, basing on the probability estimations  $s_i = \Pr(E_i)$ ,  $i = \overline{1, n}$  of  $E_i$  in  $R$  and the costs  $c_i = \text{cost}(r, E_i)$   $i = \overline{1, n}$  paid to compute the functions  $\text{test}(r, E_i)$ , we can compute the average probabilistic cost of the algorithm (3). Then, analyzing the mathematical expression  $\text{cost}(r, E)$  represented by  $c_i, s_i$   $i = \overline{1, n}$ , we try to find the best ordering  $\tau = \{\tau_1, \dots, \tau_n\}$  - a permutation of  $\{1, \dots, n\}$ , such that the value of the expression  $\text{cost}(r, E)$  is minimum.

Note that to compute test  $(r, E)$  for a given  $E$  and  $r \in R$ , the replacements in step  $i$ ) are necessary and the time cost is the same for every  $r \in R$ .

It is obvious that:

$$\text{cost}(r, E_i) = c_i > 0 \text{ (constant)} \quad \forall r \in R \quad (4)$$

Assume that for each  $E_i$ , by the rules as in §2 we can define  $s_i = \Pr(E_i)$ ,  $i = \overline{1, n}$  and the logical subexpressions are independent of each other.

The relation  $R$  is partitioned into  $T$  and  $T_i$ ,  $i = \overline{1, n}$  as follows:

$$R = T \cup \left( \bigcup_{i=1}^n T_i \right)$$

$$T = \sigma_E(R) = \{r \in R / \text{test}(r, E) = \underline{\text{true}}\}$$

$$T_i = \left. \begin{aligned} \{r \in R / \forall j, j = \overline{1, i-1} \text{ test}(r, E_j) = \underline{\text{true}}, \\ \text{test}(r, E_i) = \underline{\text{false}} \} \end{aligned} \right\}$$

it is evident that

$$T \cap T_j = \emptyset, j = \overline{1, n}$$

$$T_i \cap T_j = \emptyset, j \neq i.$$

Define

$$\rho^*(T) = \prod_{i=1}^n s_i, \quad \rho^*(T_i) = \prod_{j=1}^{i-1} s_j (1-s_i), \quad i = \overline{2, n}$$

$$\rho^*(T_1) = 1-s_1$$

We have

$$\rho^*(T) + \sum_{i=1}^n \rho^*(T_i) = \sum_{i=1}^n \prod_{j=1}^{i-1} s_j (1-s_i) + \prod_{j=1}^n s_j = 1$$

Indeed, set  $h_i = \prod_{j=1}^i s_j$ ,  $h_0 = 1$

$$\rho^*(T_i) = \prod_{j=1}^{i-1} s_j (1-s_i) = h_{i-1} - h_i, \quad i = \overline{2, n}$$

$$\rho^*(T_1) = 1-s_1 = h_0 - h_1$$

$$\sum_{i=1}^n \rho^*(T_i) + \rho^*(T) = \sum_{i=1}^n (h_{i-1} - h_i) + h_n = h_0 = 1$$

Moreover in  $T$  i.e. for the tuples  $r \in R$  for which  $\text{test}(r, E) = \underline{\text{true}}$ , it is necessary to compute all  $\text{test}(r, E_i)$ ,  $i = \overline{1, n}$  for the final result of  $\text{test}(r, E)$ , therefore it requires  $\sum_{i=1}^n c_i$ .

In  $T_i$ , because  $\text{test}(r, E_i) = \underline{\text{false}}$ , the computation of  $\text{test}(r, E)$  halts and it takes  $\sum_{j=1}^i c_j$ . By the definition 6, the average probabilistic cost of the algorithm (3) is

$$\begin{aligned} \overline{\text{cost}}_3(r, E) &= \rho^*(T) \text{cost}(r \in T, E) + \sum_{i=1}^n \rho^*(T_i) \text{cost}(r \in T_i, E) = \\ &= \left( \sum_{i=1}^n c_i \right) \prod_{i=1}^n s_i + \sum_{i=1}^n \prod_{j=2}^{i-1} s_j (1 - s_i) \left( \sum_{j=1}^i c_j \right) \end{aligned} \quad (5)$$

Return to the algorithm' (2) computing the function  $\text{test}(r, E)$ , the worst-case cost and the average probabilistic cost are the same. We have:

$$\text{cost}_2(r, E) = \sum_{i=1}^n c_i \quad (6)$$

The following result is obvious.

Proposition 1.

$$\overline{\text{cost}}_3(r, E) \leq \text{cost}_2(r, E)$$

where  $\overline{\text{cost}}_3(r, E)$  and  $\text{cost}_2(r, E)$  are expressed by the formulae (5), (6) respectively.

Proof. It is not difficult to see that:

$$\overline{\text{cost}}_3(r, E) \leq \left( \sum_{i=1}^n c_i \right) \left[ \prod_{i=1}^n s_i + \sum_{i=1}^n \prod_{j=1}^{i-1} s_j (1 - s_i) \right] = \sum_{i=1}^n c_i =$$

$$= \text{cost}_2(r, E)$$

The above proof shows that the algorithm (3) has always the cost less than the cost of algorithm (2).

Now, (5) can be transformed in the following way:

$$\overline{\text{cost}}_3(r, E) = \left( \sum_{i=1}^n c_i \right) \prod_{i=1}^n s_i + \sum_{i=1}^n (1-s_i) \prod_{j=1}^{i-1} s_j \left( \sum_{j=1}^i c_j \right)$$

Set

$$g_i = \sum_{j=1}^i c_j, g_0 = 0$$

$$\begin{aligned} \overline{\text{cost}}_3(r, E) &= g_n h_n + \sum_{i=1}^n (h_{i-1} - h_i) g_i = \\ &= g_n h_n + \sum_{i=1}^n h_{i-1} g_i - \sum_{i=1}^n h_i g_i = \\ &= \sum_{i=1}^n h_{i-1} g_i - \sum_{i=1}^{n-1} h_i g_i = \sum_{i=1}^n h_{i-1} g_i - \sum_{i=1}^n h_{i-1} g_{i-1} = \\ &= \sum_{i=1}^n h_{i-1} (g_i - g_{i-1}) = \sum_{i=1}^n c_i \prod_{j=1}^{i-1} s_j \end{aligned}$$

From here the following problem can be formulated:

Given the numbers  $c_i > 0, i = \overline{1, n}$   
 $1 \geq s_i \geq 0$

Find the best permutation  $\tau = \{\tau_1, \dots, \tau_n\}$  of  $\{1, \dots, n\}$  such that

$$A(\tau) = \sum_{i=1}^n c_{\tau_i} \prod_{j=1}^{i-1} s_{\tau_j} \rightarrow \min .$$

If it is possible to find the best permutation  $\tau$  such that  $A(\tau) \rightarrow \min$ , then by the commutativity and the associativity of



the logical operator  $\wedge$ , we have

$$\sigma_E(R) = \sigma_{\bigwedge_{i=1}^n E_i}(R) = \sigma_{\bigwedge_{i=1}^n E_{\tau_i}}(R) .$$

This transformation should allow us to calculate the function test  $(r,E)$  by the algorithm (3) not with the ordering  $\{1, \dots, n\}$  of  $E_i$  but with the ordering  $\{\tau_1, \dots, \tau_n\}$ . And so, the algorithm (3) becomes:

$$\left. \begin{array}{l} \text{Test } (r,E) := \underline{\text{false}} ; \\ \text{for } i := 1 \text{ to } n \text{ do} \\ \text{if } \neg \text{test } (r, E_{\tau_i}) \text{ then go to } L ; \\ \text{test } (r,E) := \underline{\text{true}} . \end{array} \right\} \quad (3.1)$$

L :

For this algorithm, the function test  $(r,E)$  can be computed with the average cost

$$\sum_{i=1}^n c_{\tau_i} \prod_{j=1}^{i-1} s_{\tau_j} \rightarrow \min .$$

The following proposition will show the way to find the best  $\tau$ .

Proposition 2.

Let  $s_i > 0, i = \overline{1, n}$ ,  $\tau, \tau'$  be two permutations of  $\{1, \dots, n\}$  whose  $i_0$ -th and  $(i_0+1)$ -th elements are changed with each other, i.e.

$$\tau'_{i_0} = \tau_{i_0+1}, \tau'_{i_0+1} = \tau_{i_0} .$$

If

$$t_{\tau_{i_0}} = \frac{u_{\tau_{i_0}}}{c_{\tau_{i_0}}} < t_{\tau_{i_0+1}} = \frac{u_{\tau_{i_0+1}}}{c_{\tau_{i_0+1}}}$$

$$u_i = 1 - s_i, \quad i = \overline{1, n}.$$

Then

$$A(\tau') < A(\tau)$$

Proof.

$$A(\tau) = \sum_{i=1}^n c_{\tau_i} \prod_{j=1}^{i-1} s_{\tau_j}$$

$$A(\tau') = \sum_{i=1}^n c_{\tau'_i} \prod_{j=1}^{i-1} s_{\tau'_j}$$

when  $i < i_0$

$$c_{\tau'_i} = c_{\tau_i}$$

$$\prod_{j=1}^{i-1} s_{\tau'_j} = \prod_{j=1}^{i-1} s_{\tau_j}$$

when  $i = i_0$

$$c_{\tau'_{i_0}} = c_{\tau_{i_0+1}}$$

$$\prod_{j=1}^{i_0-1} s_{\tau'_j} = \prod_{j=1}^{i_0-1} s_{\tau_j}$$

when  $i = i_0 + 1$

$$c_{\tau'_{i_0+1}} = c_{\tau_{i_0}}$$

$$\prod_{j=1}^{i_0} s_{\tau'_j} = \prod_{j=1}^{i_0-1} s_{\tau_j} \cdot s_{\tau_{i_0+1}}$$

when  $i > i_0 + 1$

$$c_{\tau'_i} = c_{\tau_i}$$

$$\prod_{j=1}^{i-1} s_{\tau'_j} = \prod_{j=1}^{i_0-1} s_{\tau'_j} \cdot s_{\tau'_{i_0}} \cdot s_{\tau'_{i_0+1}} \cdot \prod_{j=i_0+2}^{i-1} s_{\tau'_j} =$$

$$\begin{aligned}
 &= \prod_{j=1}^{i_0-1} s_{\tau_j} \cdot s_{\tau_{i_0+1}} \cdot s_{\tau_{i_0}} \cdot \prod_{j=i_0+2}^{i-1} s_{\tau_j} = \\
 &= \prod_{j=1}^{i-1} s_{\tau_j}
 \end{aligned}$$

$$A(\tau') - A(\tau) = c_{\tau_{i_0+1}} \prod_{j=1}^{i_0-1} s_{\tau_j} + c_{\tau_{i_0}} \prod_{j=1}^{i_0-1} s_{\tau_j} \cdot s_{\tau_{i_0+1}}$$

$$- c_{\tau_{i_0}} \prod_{j=1}^{i_0-1} s_{\tau_j} - c_{\tau_{i_0+1}} \prod_{j=1}^{i_0-1} s_{\tau_j} \cdot s_{\tau_{i_0}}$$

$$= \prod_{j=1}^{i_0-1} s_{\tau_j} (c_{\tau_{i_0+1}} + c_{\tau_{i_0}} s_{\tau_{i_0+1}} - c_{\tau_{i_0}} - c_{\tau_{i_0+1}} s_{\tau_{i_0}})$$

$$= \prod_{j=1}^{i_0-1} s_{\tau_j} (c_{\tau_{i_0+1}} (1 - s_{\tau_{i_0}}) - c_{\tau_{i_0}} (1 - s_{\tau_{i_0+1}}))$$

$$= \prod_{j=1}^{i_0-1} s_{\tau_j} \cdot c_{\tau_{i_0}} \cdot c_{\tau_{i_0+1}} \left( \frac{1 - s_{\tau_{i_0}}}{c_{\tau_{i_0}}} - \frac{1 - s_{\tau_{i_0+1}}}{c_{\tau_{i_0+1}}} \right) < 0$$

$$\rightarrow A(\tau') < A(\tau).$$

Proposition 3.

Given the numbers

$$\left. \begin{aligned}
 0 &\leq s_i \leq 1 \\
 0 &< c_i
 \end{aligned} \right\} i = \overline{1, n}$$

Set

$$t_i = \frac{1-s_i}{c_i}$$

If  $\tau_0 = \{1, \dots, n\}$  satisfies  $t_i \geq t_{i+1}$ ,  $i = \overline{1, n-1}$  then  $A(\tau_0)$  is the minimum.

Proof.

Let  $\tau = \{\tau_1, \dots, \tau_n\}$  be a permutation of  $\{1, \dots, n\} = \tau_0$ . We have to prove that  $A(\tau_0) \leq A(\tau)$ .

First we remark that from  $\{t_{\tau_i}\}_{i=1}^n$   $\otimes$  the sequence  $\{t_i\}_{i=\overline{1, n}}$  (corresponding to  $\tau_0$ ) can be obtained by

- (i) the bubble sorting algorithm permuting sequentially the adjacent elements  $t_{\tau_{i_0}}$ ,  $t_{\tau_{i_0+1}}$  satisfying  $t_{\tau_{i_0}} < t_{\tau_{i_0+1}}$  and
- (ii) the permutations (if necessary) of the elements with equal values in the obtained sequence.

By proposition 2, if (i) should be carried out then we should obtain  $\tau^1$  satisfying  $A(\tau^1) < A(\tau)$ .

Basing upon the proof of the proposition 2, we have: The permutations of the elements with equal values in the sequence  $\{t_{\tau_i}\}_{i=1, n}$  do not change the value of  $A$ , i.e.  $A(\tau_0) = A(\tau^1)$ .

From here follows  $A(\tau_0) = A(\tau^1) < A(\tau)$ . When the step (i) does not take places, it is not difficult to see that  $A(\tau_0) = A(\tau)$ .

The following algorithm will give the best result for any conjunctive logical expression.

Algorithm A1.

Input:  $E = E_1 \wedge \dots \wedge E_n .$

$$R = \{r : U \rightarrow \prod_{i=1}^k D_i / \forall i r(A_i) \in D_i\}$$

Output:  $\tau = \{\tau_1, \dots, \tau_n\}$  is the permutation of  $\{1, \dots, n\}$  such that

$$A(\tau) = \sum_{i=1}^n c_{\tau_i} \prod_{j=1}^{i-1} s_{\tau_j} \rightarrow \min .$$

Method

- 1) For each  $i$ , estimate the probability  $\Pr(E_i)$  by the formulae in §2 or by the formulae given by the system programmers basing upon the statistical parameters during the manipulation of  $R$ .
- 2) If there exists  $i_0$  such that  $\Pr(E_{i_0}) = 0$ , then inform  $\sigma_E(R) = \emptyset$ .
- 3) If there exist  $i_0$  such that  $\Pr(E_{i_0}) = 1$  then delete  $E_{i_0}$  from  $E$ .

More generally, denote  $I = \{i_0 / S(E_{i_0}) = 1\}$ . Consider

$$E = \bigwedge_{i \in \{1, \dots, n\} - I} E_i$$

Renumber the expressions  $E_i$  in

$$E = \bigwedge_{i \in \{1, \dots, n\} - I} E_i$$

$$n := n - \text{card } I$$

- 4) Define  $c_i$ ,  $i=\overline{1,n}$  (In practice, in order to define  $c_i$ , we compute the function test  $(r, E_i)$  for any tuple and give  $c_i = \text{cost}(r, E_i)$ ).

For instance: if

$$E_i = A\theta B \quad \text{then} \quad c_i = 2a + b$$

$$E_i = A\theta c, \quad c_i = a + b$$

$$E_i = c\theta A, \quad c_i = a + b$$

where

a is the cost to substitute A by r.A;

b is the cost paid to compare two elements in  $\bigcup_{i=1}^k D_i$ .

- 5)  $u_i = 1 - s_i$ ,  $i = \overline{1, n}$ .
- 6)  $t_i = u_i / c_i$ ,  $i = \overline{1, n}$ .
- 7) Sort  $\{t_i\}$  such that  $t_i \geq t_{i+1}$   $i = \overline{1, n-1}$   
 Step 7 can be performed by one of the sorting algorithms, in general, of complexity  $O(n \log n)$ .
- 8) Print the best ordering obtained  $\tau = \{\tau_1, \dots, \tau_n\}$ .
- 9) Print the value  $A(\tau) = \sum_{i=1}^n c_{\tau_i} \prod_{j=1}^{i-1} s_{\tau_j}$ .

Costing the algorithm A1.

Steps 1,2,3	are performed with the cost	$K_1 n$
step 4	has the complexity	$K_2 n$
step 5,6		$K_3 n$
step 7		$K_4 n \log n$
step 8,9		$K_5 n$

The complexity of the algorithm A1 is of

$$\begin{aligned}Kn + H n \log n &\approx O(n \log n) \\K &= K1 + K2 + K3 + K5 \\H &= K4 .\end{aligned}$$

Remark 4.

The proof of the proposition 3 is based on the bubble sorting algorithm of complexity  $O(n^2)$  but the step 7 of the algorithm A1 uses any sorting algorithm of complexity  $O(n \log n)$ . However, there is no matters about the correctness of the algorithm A1.

The algorithm A1 can be implemented without any access to the secondary memory devices containing the file R.

Theorem 1.

Let R be a relation,  $\text{card } R = N$ ,  $E = \bigwedge_{i=1}^n E_i$ ,  $\tau_0 = \{1, \dots, n\}$  is the best ordering of  $E_i$ 's i.e.

$$A(\tau_0) = \sum_{i=1}^n c_i \prod_{j=1}^{i-1} s_j \rightarrow \min$$

Then, the cost of the algorithm (1) finding  $\sigma_E(R)$  with the function test  $(r, E)$  computed by:

1) Algorithm (2) is  $C^1 = N \cdot \sum_{i=1}^n c_i$

2) Algorithm (3) with the best ordering  $\tau_0$  of  $E_i$ 's is

$$C^2 = N \cdot \sum_{i=1}^n c_i \prod_{j=1}^{i-1} s_j + F(n)$$

where  $F(n) = Kn + Hn \log n$  is the cost paid to perform the algorithm A1.

3) Algorithm (3) with an arbitrary ordering  $\tau = \{\tau_1, \dots, \tau_n\}$  is

$$C^3 = N \sum_{i=1}^n C_{\tau_i} \prod_{j=1}^{i-1} s_{\tau_j}$$

we have the inequalities:

$$\begin{aligned} C^1 &> C^3 \\ C^1 &> C^2 \quad \text{with large } N \\ C^3 &> C^2 \quad \text{with large } N. \end{aligned}$$

#### §4 Extensions

Extension 1. If  $E$  is of the form  $E = E_1 \vee \dots \vee E_n$  then using the symbols as above and the De Morgan's law

$$\neg(E_1 \vee \dots \vee E_n) = \neg E_1 \wedge \dots \wedge \neg E_n$$

we have: the cost payed to compute test  $(r, E)$  is

$$\overline{\text{cost}}(r, E) = \sum_{i=1}^n c_i \prod_{j=1}^{i-1} s'_j \quad \text{where } s'_j = 1 - s_j, \quad j = \overline{1, n}$$

#### Proposition 4.

$$\begin{aligned} \text{Given} \quad s_i &= \text{Pr}(E_i) & 0 \leq s_i \leq 1 \\ & & c_i > 0 & i = \overline{1, n} \\ t_i &= s_i / c_i \end{aligned}$$

If  $\tau_0 = \{1, \dots, n\}$  satisfies  $t_i > t_{i+1}, i = \overline{1, n-1}$  then

$$\text{cost}(r, E) = \sum_{i=1}^n c_i \prod_{j=1}^{i-1} s'_j \rightarrow \min.$$



Extension 2.

Algorithm A2.

Input: An arbitrary logical expression E (as defined by def.2).

Output The best ordering of the simple logical sub-expressions of E.

Method.

1) Reduce E to the conjunctive disjunctive normal form

$$r(E) = \bigvee_{i=1}^n \bigwedge_{j=1}^{n_i} E_j^i .$$

2) Apply algorithm A1 to

$$E_i = \bigwedge_{j=1}^{n_i} E_j^i$$

to give the best ordering  $\tau^i$  of  $E_j^i$ ,  $j=\overline{1, n_i}$ .

3) Apply the modified algorithm to  $\bigvee_{i=1}^n E_i$  with  $c_i = A(\tau^i)$  and  $s_i = \text{Pr}(E_i)$  defined by the estimating formulae analogous to one's in §2.

4) Print the best ordering  $E = \bigvee_{i=1}^n \bigwedge_{j=1}^{n_i} E_{\tau_j^i}^i$ .

5) Print the value  $C = \sum_{i=1}^n c_{\tau_i} \prod_{j=1}^{i-1} s_{\tau_j}$ .

## CONCLUSION

Independently, our approach is quite near to the Hanani's one [5]. However, our approach seems to be more straightforward, easy for extensions and the complexity analysis of the algorithm proposed is much elaborate.

## ACKNOWLEDGEMENT

The authors wish to thank Dr Béla Uhrin and Katalin Fridl for their valuable comments and suggestions.

## REFERENCES

- [1] Ulmann, J.; Principles of database systems. Computer Science Press. Second edition, 1982.
- [2] Adiba, M. and Delobel, C.: Bases de données et systèmes relationnels. Paris, Dunod, 1982.
- [3] Steve Talbot: An investigation into logical optimization of relational query languages. The comp. J. Vol. 27, No.4, November 1984.
- [4] Matthias Jarke and Jürgen Koch: Query optimization in Database systems. ACM Computing Surveys, Vol.16. No.2, June 1984.
- [5] Hanani, M.Z.: An optimal evaluation of Boolean expression in an on line query system. Comm. ACM 20, 5, May 1977.

A relációs kifejezések kiválasztásának optimalizálásáról

J. DEMETROVICS, HO THUAN, NGUEN THANH THUY

Összefoglaló

Legyen  $E = \bigwedge_{i=1}^n E_i$  egy feltételes kifejezés és  $R$  egy

reláció. A cikkben egy  $O(n \log n)$  algoritmust mutatnak be a szerzők, amely a  $\sigma_E(R) := \{r \in R / r \text{ kielégíti az } E\}$  mennyiséget /átlagban/ minimális lépésszámban határozza meg /azaz, amely komplexitásának várható értéke minimális/.

Оптимизация выборок из реляционных выражений.

Й. Деметрович, Хо Тхуан, Нгуен Тханх Тхуй

Резюме

Пусть  $E = \bigwedge_{i=1}^n E_i$  есть условное выражение и  $R$  реляция.

В статье показывается  $O(n \log n)$  алгоритм который /в среднем/ минимизирует число шагов для нахождения  $\sigma/R := \{r \in R / r \text{ исполняет } E\}$ .



BALANCED RELATION SCHEME  
AND THE PROBLEM OF KEY REPRESENTATION

*J. DEMETROVICS, HO THUAN*

Computer and Automation Institute, Hungarian  
Academy of Sciences

*NGUYEN XUAN HUY, LE VAN BAO*

Institute of Computer Sciences and Cybernetics  
Center of Scientific Research of Vietnam

ABSTRACT

In this paper the concepts of so-called balanced relation scheme and the operator of translation of relation scheme are presented.

Basing on this special type of relation schemes and this operator, representation of all keys of any relation scheme is given.

50. INTRODUCTION

The relational data model was first introduced in Codd [1] and new become a most promising one. The study of dependency structure in relational databases [2] and the so-called generalized functional dependencies [3,8] give us a powerful tool to deal with these dependencies. The concept of translation of relation scheme [4] seems be useful in the sense that its reduces a relation scheme to a simpler one.

The problem of key representation was investigated in [5] by Bekessy A. and Demetrovics J. in connection with some estimations of the upper bound of the cardinality of the set of all keys of a relation scheme.

In this paper we present the main results about the translation of relation scheme and consider the so-called balanced

relation scheme. In connection with these results, the problem of key representation is solved.

The notation used here is the same as in [6].

## §1. TRANSLATION OF RELATION SCHEME

Definition 1.1. Let  $S = \langle \Omega, F \rangle$  be a relation scheme, where

$$\Omega = \{A_1, A_2, \dots, A_n\}$$

is the set of attributes,

$$F = \{L_i \rightarrow R_i \mid L_i, R_i \subseteq \Omega; i=1, 2, \dots, k\}$$

is the set of functional dependences, and  $Z \subseteq \Omega$ , be an arbitrary subset of  $\Omega$ . We define a new relation scheme  $\tilde{S} = \langle \tilde{\Omega}, \tilde{F} \rangle$  by:

$$\tilde{\Omega} = \Omega \setminus Z \quad (= \bar{Z})$$

$$\tilde{F} = \{L_i \setminus Z \rightarrow R_i \setminus Z \mid (L_i \rightarrow R_i) \in F, i=1, 2, \dots, k\}$$

Then  $\tilde{S}$  is said to be obtained from  $S$  by a  $Z$ -translation, and the notation

$$\tilde{S} = S - Z$$

is used.

From the above definition, it is clear that, after the transformation,  $\tilde{F}$  can contain the functional dependencies of the following forms:

- (i)  $\emptyset \rightarrow \emptyset$ ;
- (ii)  $X \rightarrow \emptyset$  where  $X \subseteq \tilde{\Omega}$ ;  $X \neq \emptyset$ ;
- (iii)  $\emptyset \rightarrow X$  where  $X \subseteq \tilde{\Omega}$ ,  $X \neq \emptyset$ .

However, by the algorithm to find the closure  $X^+$  of the subset  $X \subseteq \Omega$ , w.r.t.  $F$  [7] we observe that the omission of func-

tional dependencies of the form (i) and (ii) in  $\tilde{F}$  do not change  $\mathcal{K}_{\tilde{S}}$ , the set of all keys of  $\tilde{S}$ .

Definition 1.2. Let  $S = \langle \Omega, F \rangle$  be a relation scheme, and  $\mathcal{K}_S$  be the set of all keys of  $S$ . We define a partition of  $\Omega$  as follow:

$$\Omega = G \cup \Omega^{(1)} \cup \Omega^{(0)}$$

where

$$G = \bigcap_{X_i \in \mathcal{K}_S} X_i;$$

$$\Omega^{(1)} = \bigcup_{X_i \in \mathcal{K}_S} X_i \setminus G = H \setminus G;$$

$$\Omega^{(0)} = \Omega \setminus H.$$

Sometimes, for the sake of simplicity, the notation

$$\Omega = G | \Omega^{(1)} | \Omega^{(0)} = H | \Omega^{(0)}$$

is also used.

Definition 1.3. Let  $\Omega$  be the universe of attributes,  $X \subseteq \Omega$ ,  $\mathcal{M} \subseteq 2^\Omega$ ,  $\mathcal{N} \subseteq 2^\Omega$ . We define

$$X \oplus \mathcal{M} = \{XY \mid Y \in \mathcal{M}\}$$

$$\mathcal{M} \oplus \mathcal{N} = \{YZ \mid Y \in \mathcal{M}, Z \in \mathcal{N}\}$$

Here  $XY$  means  $X \cup Y$ .

The next proposition will be needed in the sequel.

Proposition 1.1. Let  $S = \langle \Omega, F \rangle$  be a relation scheme,  $X, Y \subseteq \Omega$ , then

$$((X)^+ Y)^+ = (X(Y)^+)^+ = (XY)^+ \tag{1}$$

where  $X^+$  is the closure of  $X$  w.r.t.  $F$ .

Proof. It is sufficient to prove that

$$(X^+Y)^+ = (XY)^+$$

By the definition of the closure  $X^+$  of  $X$ , it is obvious that

$$X^+ \supseteq X$$

Hence

$$X^+Y \supseteq XY.$$

By the algorithm to find the closure, we have

$$(X^+Y)^+ \supseteq (XY)^+. \tag{2}$$

On the other hand, from

$$X \xrightarrow{*} X^+$$

we have

$$XY \xrightarrow{*} X^+Y$$

or equivalently:

$$X^+Y \subseteq (XY)^+ \quad (\text{c.f. [7]})$$

Hence

$$(X^+Y)^+ \subseteq ((XY)^+)^+ = (XY)^+ \tag{3}$$

Combining (2) with (3) we obtain (1). The proof is complete.

We are now in a position to prove the following theorem, characterizing the fundamental property of translation of relation scheme.

Theorem 1.1. Let  $S = \langle \Omega, F \rangle$  be a relation scheme, and  $Z \subseteq \Omega$ . If  $\tilde{S} = S - Z = \langle \bar{Z}, \tilde{F} \rangle$ , then for every  $X \subseteq \bar{Z}$  we have



$$Z(X)_{\tilde{S}}^+ = (ZX)_{\tilde{S}}^+ \quad (*) \quad (4)$$

Formula (4) expresses the relation between closures in the source relation scheme  $S$  and the target one  $\tilde{S}$ .

Proof. First, observe that, by the definition of the closure of a set of attributes, from  $X \subseteq \tilde{Z}$  we have

$$(X)_{\tilde{S}}^+ \subseteq Z \quad \text{and therefore}$$

$$(X)_{\tilde{S}}^+ \cap Z = \emptyset$$

Now we prove the theorem 1 by induction on  $h$ , the step number for constructing the sequences  $\{(X)_{\tilde{S}}^{(h)}\}$  and  $\{(ZX)_{\tilde{S}}^{(h)}\}$ ,  $h=0,1,2,\dots$  by the algorithm computing the closure  $X^+$  of  $X$ .

Basis:  $h=0$ . It is clear that

$$X_{\tilde{S}}^{(0)} = X \quad \text{and} \quad (ZX)_{\tilde{S}}^{(0)} = ZX.$$

$$\text{Therefore: } Z(X)_{\tilde{S}}^{(0)} = (ZX)_{\tilde{S}}^{(0)} = ZX.$$

Induction: Let  $h > 0$  and assume that

$$Z(X)_{\tilde{S}}^{(h)} = (ZX)_{\tilde{S}}^{(h)} \quad (5)$$

We shall prove that

$$Z(X)_{\tilde{S}}^{(h+1)} = (ZX)_{\tilde{S}}^{(h+1)} \quad (6)$$

For that, it is sufficient to prove that, when processing from step  $h$  to step  $(h+1)$ , the sets  $(X)_{\tilde{S}}^{(h)}$  and  $(ZX)_{\tilde{S}}^{(h)}$  are added by the same elements i.e.

$$(X)_{\tilde{S}}^{(h+1)} \setminus (X)_{\tilde{S}}^{(h)} = (ZX)_{\tilde{S}}^{(h+1)} \setminus (ZX)_{\tilde{S}}^{(h)} \quad (7)$$

(\*) Sometimes, we omit the subscript  $S$  (or  $\tilde{S}$ ) if the context is obvious.

Let  $P, Q$  denote the left and right side of (7) respectively. We find that  $x \in P$  if and only if  $\exists (L_i \setminus Z \rightarrow R_i \setminus Z) \in \tilde{F}$  such that

$$L_i \setminus Z \subseteq (X)_{\tilde{S}}^{(h)}, \quad x \in R_i \setminus Z \quad \text{and} \quad x \notin (X)_{\tilde{S}}^{(h)}.$$

So, we have

$$L_i \subseteq Z(X)_{\tilde{S}}^{(h)} \quad \text{and by (5):} \quad L_i \subseteq (ZX)_{\tilde{S}}^{(h)}.$$

Since  $x \in R_i \setminus Z$  then  $x \notin Z$  and  $x \in R_i$ . On the other hand, since  $x \notin (X)_{\tilde{S}}^{(h)}$  then  $x \notin Z(X)_{\tilde{S}}^{(h)}$ . Again, by (5) we have:

$$x \notin (ZX)_{\tilde{S}}^{(h)}.$$

From  $L_i \subseteq (ZX)_{\tilde{S}}^{(h)}$ ,  $x \in R_i$ ,  $x \in (ZX)_{\tilde{S}}^{(h)}$ , then, by the algorithm to find the closure, we have  $x \in Q$ .

In the inverse direction, if  $y \in Q$ , that is  $\exists (L_i \rightarrow R_i) \in F$  such that  $L_i \subseteq (ZX)_{\tilde{S}}^{(h)}$ ,  $y \in R_i$  and  $y \notin (ZX)_{\tilde{S}}^{(h)}$ . Consequently, by (5) we have  $L_i \subseteq (ZX)_{\tilde{S}}^{(h)}$  and  $y \notin Z(X)_{\tilde{S}}^{(h)}$ .

Hence

$$y \notin (X)_{\tilde{S}}^{(h)} \quad \text{and} \quad y \notin Z.$$

Therefore  $y \in R_i \setminus Z$  and  $L_i \setminus Z \subseteq (X)_{\tilde{S}}^{(h)}$ .

Basing again on the algorithm to find the closure, we find that  $y \in P$ .

The equality (7) is proved.

It is obvious that

$$Z \cap (X)_{\tilde{S}}^{(h)} = \emptyset \quad \text{and} \quad Z \cap (X)_{\tilde{S}}^{(h+1)} = \emptyset.$$

From (7) we have:

$$Z(X)_{\tilde{S}}^{(h+1)} \setminus Z(X)_{\tilde{S}}^{(h)} = (ZX)_{\tilde{S}}^{(h+1)} \setminus (ZX)_{\tilde{S}}^{(h)}$$

Basing on (5), we obtain

$$Z(X)_{\tilde{S}}^{(h+1)} = (ZX)_S^{(h+1)}$$

showing that:

$$Z(X)_{\tilde{S}}^+ = (ZX)_S^+.$$

Let us denote by  $\mathcal{K}_S$  and  $\mathcal{K}_{\tilde{S}}$  the set of all keys of  $S$  and the set of all keys of  $\tilde{S}$  respectively, where  $S = \langle \Omega, F \rangle$ ,  $\tilde{S} = S - Z$ ,  $Z \subseteq \Omega$ .

We have the following lemma.

Lemma 1.1. If  $X$  is a superkey of  $S$ , then  $X \setminus Z$  is a superkey of  $\tilde{S}$  and conversely if  $Y$  is a superkey of  $\tilde{S}$ , then  $YZ$  is a superkey of  $S$ .

Proof.

1. Since  $X$  is a superkey of  $S$ ,

$$(X)_S^+ = \Omega = Z \bar{Z}.$$

Put  $V = X \setminus Z$ , it is obvious that  $X \subseteq VZ$  and  $V \subseteq \bar{Z}$ .

Hence  $X_S^+ \subseteq (VZ)_S^+ = \Omega$ . (because  $X_S^+ = \Omega$ )

By the just proved theorem we have

$$Z(V)_{\tilde{S}}^+ = (ZV)_S^+$$

Consequently:

$$Z(V)_{\tilde{S}}^+ = \Omega = Z \bar{Z}.$$

showing that  $(V)_{\tilde{S}}^+ = \bar{Z}$ . (since  $Z \bar{Z} = \emptyset$  and  $(V)_{\tilde{S}}^+ \subseteq \bar{Z}$ ) or in other words,  $V$  is a superkey of  $\tilde{S}$ .

2. Conversely, if  $Y$  is a superkey of  $\tilde{S}$ , i.e.,

$$(Y)_{\tilde{S}}^+ = \bar{Z} (= \Omega \setminus Z).$$

Then, by theorem 1.1, we have:

$$(ZY)_S^+ = Z(Y)_{\tilde{S}}^+ = Z \bar{Z} = \Omega$$

showing that  $ZY$  is a superkey of  $S$ .

Remark 1.1. Both Theorem 1.1. and lemma 1.1. can also be proved by using the fundamental lemma 1.1. in [4] which gives us another characterization of translation of relation scheme.

Corollary 1.1. If  $\tilde{S} = S - Z$  then:

- (i)  $\mathcal{K}_S = \mathcal{K}_{\tilde{S}} \leftrightarrow Z \subseteq \Omega^{(0)}$ ;
- (ii)  $\mathcal{K}_S = Z \oplus \mathcal{K}_{\tilde{S}} \leftrightarrow Z \subseteq G$ .

Proof. (i)  $\rightarrow$  If  $\mathcal{K}_S = \mathcal{K}_{\tilde{S}}$  and suppose there exist an attribute  $A_i \in Z \setminus \Omega^{(0)}$ , i.e.  $A_i$  is a prime attribute ( $A_i \in H$ ).

Then  $\exists K \in \mathcal{K}_S$  such that  $A_i \in K$ .

Since  $\mathcal{K}_S = \mathcal{K}_{\tilde{S}}$ , so  $K \in \mathcal{K}_{\tilde{S}}$  i.e.  $A_i$  is also a prime attribute of  $\tilde{S}$ . We thus arrive to a contradiction.

Consequently  $Z \setminus \Omega^{(0)} = \emptyset$ , or  $Z \subseteq \Omega^{(0)}$ .

$\leftarrow$  First, observe that if  $X$  is a superkey of  $S$  then after removing non prime attributes from  $X$ , the remaining part of  $X$  is also a superkey of  $S$ . In other words, if  $X$  is a superkey of  $S$  then with all  $Z \subseteq \Omega^{(0)}$ ,  $X' = X \setminus Z$  is also a superkey of  $S$ .

Suppose that  $Z \subseteq \Omega^{(0)}$ , we shall prove that  $\mathcal{K}_S = \mathcal{K}_{\tilde{S}}$ .

Indeed, if  $K \in \mathcal{K}_S$  then in particular,  $K$  is a superkey of  $S$ . By lemma 1.1,  $K \setminus Z$  is a superkey of  $\tilde{S}$ .

Moreoether, it is clear that  $Z \cap K = \emptyset$ .

Consequently  $K \setminus Z = K$ , i.e.  $K$  is a superkey of  $\tilde{S}$ .

Now, suppose that  $K' \subseteq K$  and  $K'$  is a key of  $\tilde{S}$ . Again, by lemma 1.1,  $ZK'$  is a superkey of  $S$ .

Removing from  $ZK'$  the subset  $Z \subseteq \Omega^{(0)}$ , we find that  $K'$  is a superkey of  $S$  too. Since  $K$  is a key of  $S$ , we conclude that  $K' = K$ , showing  $K \in \mathcal{K}_{\tilde{S}}$ .

Hence  $\mathcal{K}_S \subseteq \mathcal{K}_{\tilde{S}}$ .

Conversely, given  $K \in \mathcal{K}_{\tilde{S}}$ .

Obviously  $K \cap Z = \emptyset$ .

By Lemma 1.1,  $ZK$  is a superkey of  $S$ .

Removing from  $ZK$  the set  $Z \subseteq \Omega^{(0)}$ , we find that  $K$  is a superkey of  $S$ .

Now suppose that  $K' \subseteq K$  and  $K'$  is a key of  $S$ .

It is easy to see that  $K' \cap Z = \emptyset$ , so  $K' \setminus Z = K'$ .

Once again, by Lemma 1.1,  $K'$  is a superkey of  $\tilde{S}$ .

Since  $K$  is a key of  $\tilde{S}$ , so  $K' = K$  showing that  $K \in \mathcal{K}_S$ . Hence  $\mathcal{K}_{\tilde{S}} \subseteq \mathcal{K}_S$ . So (i) is proved.

(ii)  $\rightarrow$  If  $\mathcal{K}_S = Z \oplus \mathcal{K}_{\tilde{S}}$  then obviously  $Z$  is contained in every key of  $S$ .

Hence  $Z \subseteq G$ .

$\leftarrow$  Suppose that  $Z \subseteq G$ . We shall show that every element  $K \in \mathcal{K}_S$  can be represented in the form  $K = Z\tilde{K}$ , where  $\tilde{K} \in \mathcal{K}_{\tilde{S}}$ , and, in the inverse direction, for every element  $\tilde{K} \in \mathcal{K}_{\tilde{S}}$  we can show that  $Z\tilde{K} \in \mathcal{K}_S$ .

Indeed, if  $K \in \mathcal{K}_S$  then from  $Z \subseteq G$ , we have  $Z \subseteq K$ , i.e.,  $K = Z\tilde{K}$ ,  $Z \cap \tilde{K} = \emptyset$ .

By the lemma 1.1,  $\tilde{K} = K \setminus Z$  is a superkey of  $\tilde{S}$ .

Now suppose there exist  $\tilde{K}' \subseteq \tilde{K}$  and  $\tilde{K}'$  is a key of  $\tilde{S}$ . Then, by Lemma 1.1,  $Z\tilde{K}'$  is a superkey of  $S$  and obviously  $Z\tilde{K}' \subseteq Z\tilde{K} = K$ .

Since  $K$  is a key of  $S$ , one must have

$$Z\tilde{K}' = K = Z\tilde{K}.$$

Hence  $\tilde{K}' = \tilde{K}$ , i.e.  $\tilde{K} \in \mathcal{K}_{\tilde{S}}$ .

Conversely, if  $\tilde{K} \in \mathcal{K}_{\tilde{S}}$  then by Lemma 1.1,  $K = Z\tilde{K}$  is a superkey of  $S$ .

Suppose that there exists  $K' \subset K$  and  $K'$  is a key of  $S$ . Since  $Z$  is contained every key of  $S$ , so  $Z \subseteq K'$ .

Put  $\tilde{K}' = K' \setminus Z$ , we obtain  $K' = Z\tilde{K}'$ . By Lemma 1.1,  $\tilde{K}'$  is a superkey of  $\tilde{S}$ .

From  $Z\tilde{K}' = K' \subset K = Z\tilde{K}$  and  $Z \cap \tilde{K}' = Z \cap \tilde{K} = \emptyset$ , we deduce that  $\tilde{K}' \subset \tilde{K}$ . We arrive to a contradiction. Hence  $K' = K$ , i.e.  $K \in \mathcal{K}_S$ .

(ii) is proved and the proof of corollary 1.1. is complete.

Definition 1.4. An attribute  $A_j \in \Omega$  is said to be a deterministic one if, for every  $(L_i \rightarrow R_i) \in F$ ,  $A_j \in R_i$  implies  $A_j \in L_i$ . In other words,  $A_j$  is a deterministic attribute iff when it belongs to the right side of some functional dependency, it must also belong to the left side of this FD.

Let be given the relation scheme

$$S = \langle \Omega, F \rangle$$

where

$$\Omega = \{A_1, A_2, \dots, A_n\}$$

$$F = \{L_i \rightarrow R_i, i=1, 2, \dots, k\}$$

As in [6] we denote

$$L = \bigcup_{i=1}^k L_i, \quad R = \bigcup_{i=1}^k R_i.$$

Following the definition 1.4, it is obvious that if  $\Omega \setminus (L \cup R) \neq \emptyset$  then all elements of this subset are deterministic attributes.

The following theorem establishes the relation between the set of deterministic attributes and  $G$  - the intersection of all keys of  $S$ .

Theorem 1.2. The three following sets are equal:

$$D = G = T$$

where  $D$  is the set of all deterministic attributes of  $S$ ;

$G = \bigcap_{K \in \mathcal{K}_S} K$  - the intersection of all keys of  $S$ ;

$$T = \Omega \setminus \bigcup_{i=1}^k (R_i \setminus L_i)$$

Proof.

1.  $D \subseteq G$ .

Suppose that  $x \in D$  and there is a key  $K$  such that  $x \notin K$ . Since

$(K)_S^+ = \Omega$ , so  $x \in (K)_S^+$ . By the algorithm to find the closure of a subset of attributes w.r.t.  $F$ , there exist  $h$  and some  $(L_i \rightarrow R_i) \in F$  such that

$$L_i \subseteq (K)_S^{(h)}, x \notin L_i, x \in R_i$$

This contradicts the fact that  $x$  is a deterministic attribute.

Hence,  $x \in D$  implies  $x \in K, \forall K \in S$ . In other words,  $x \in G$ .

2.  $G \subseteq D$

Suppose  $x \in G$  and there exists  $(L_i \rightarrow R_i) \in F$  such that  $x \in R_i$  but  $x \notin L_i$ .

Consider the set  $M = \Omega \setminus \{x\}$

Since  $x \notin L_i$ , it is clear that  $L_i \subseteq M$ . From  $M \xrightarrow{*} L_i, L_i \rightarrow R_i, R_i \xrightarrow{*} \{x\}$  we have  $M \xrightarrow{*} \{x\}$

Combining with  $M \xrightarrow{*} M$ , we obtain

$$M \xrightarrow{*} M \cup \{x\} = \Omega,$$

showing that  $M$  is a superkey of  $S$ . Let  $K$  be a key contained in  $M$ .

From  $x \notin M$ , we have  $x \notin K$ .

This contradicts the fact that  $x \in G$ .

Consequently, if  $x \in G$  then from  $(L_i \rightarrow R_i) \in F$  and  $x \in R_i$  we must have  $x \in L_i$ , i.e.  $x$  is a deterministic attribute.

3.  $D \subseteq T$ .

If  $x \in D$ , so by the definition 1.4.,  $x \notin R_i \setminus L_i$  for every  $i=1, 2, \dots, k$ , i.e.  $x \notin \bigcup_{i=1}^k (R_i \setminus L_i)$ .

$$\text{Hence } x \in \Omega \setminus \bigcup_{i=1}^k (R_i \setminus L_i) = T.$$

4.  $T \subseteq D$

If  $x \in T = \Omega \setminus \bigcup_{i=1}^k (R_i \setminus L_i)$ , then  $\forall i=1, 2, \dots, k, x \notin (R_i \setminus L_i)$ .

It means that if  $x \in R_i$  then  $x \in L_i$ , i.e.  $x$  is a deterministic attribute,  $x \in D$ .

The proof of theorem 1.2. is complete.

Remark 1.2. In the case  $L_i \cap R_i = \emptyset, \forall i=1, 2, \dots, k$  we have:

$$G = \Omega \setminus \bigcup_{i=1}^k (R_i \setminus L_i) = \Omega \setminus R.$$

This result has been proved in [6] by an another way.

Until now, in the published litterature, we have not the explicit expression for the set  $\Omega^{(0)}$  (equivalently, for  $H$  - the union of all keys of  $S$ ).

However in [6] it is shown that

$$R' = R \setminus L \subseteq \Omega^{(0)}.$$

Moreover, we have

Lemma 1.2.

$$R'' = \bigcup_{L_i \subseteq G} (R_i \setminus L_i) \subseteq \Omega^{(0)}.$$

Proof. If  $x \in R''$  then  $\exists (L_i \rightarrow R_i) \in F$  such that  $x \in R_i$  and  $x \notin L_i$ .  
Let  $K$  be an arbitrary key of  $S$  ( $K \in S$ ).

We shall show that  $x \notin K$ .

Since  $L_i \subseteq G$ , so  $L_i \subseteq K$ . Suppose that  $x \in K$ .

Then from  $x \notin L_i$  and  $L_i \subseteq K$ , we have

$$L_i \subseteq K \setminus \{x\} = K'$$

Obviously:

$$L_i \rightarrow R_i \xrightarrow{*} \{x\} \quad (x \in R_i)$$

$$K' \xrightarrow{*} L_i$$

Consequently

$$K' \xrightarrow{*} \{x\}.$$



Combining with  $K' \rightarrow K'$ , we have

$$K' \xrightarrow{*} K' \{x\} = K$$

This contradicts the fact that  $K$  is a key.

Hence,  $\forall K \in \mathcal{K}_S : x \notin K$ , i.e.  $x \in \Omega^{(o)}$ .

Corollary 1.2.

$$\bigcup_{L_i = \emptyset} R_i \subseteq \bigcup_{L_i \subseteq G} (R_i \setminus L_i) \subseteq \Omega^{(o)}.$$

The proof is obvious.

This corollary shows that we can eliminate from a relation scheme all FD of the form  $\emptyset \rightarrow R_i$ , while preserving its set of all keys.

The following lemma gives us a constructive way for extending a given subset of  $\Omega^{(o)}$ .

Lemma 1.3. For every  $X \subseteq G$ ,  $Y \subseteq \Omega^{(o)}$ , we have  $(XY)_S^+ \setminus X \subseteq \Omega^{(o)}$ .

Proof. If  $x \in (XY)_S^+ \setminus X$ , so  $x \in (XY)_S^+$  and  $x \notin X$ .

Suppose that  $x \in \Omega^{(o)}$ . Obviously  $x \notin Y$ .

Since  $x \in (XY)_S^+$ , so  $XY \xrightarrow{*} \{x\}$ .

From  $x \notin X$ ,  $x \notin Y$ , then  $x \notin XY$ .

Since  $x \in \Omega^{(o)}$ , there exists a key  $K \in \mathcal{K}_S$  such that  $x \in K$ .

Let  $K' = K \setminus \{x\}$ , ( $K' \subseteq K$ )

It is clear that  $XYK' \xrightarrow{*} K' \{x\} = K$  showing  $XYK'$  is a superkey of  $S$ .

Let us eliminate from  $XYK'$  the subset  $Y \subseteq \Omega^{(o)}$ , giving  $XK'$  is a superkey of  $S$ . On the other hand from  $X \subseteq G \subseteq K$ ,  $x \notin X$  we have:

$$K' = K \setminus \{x\} \supseteq X,$$

showing that  $XK' = K'$  is a superkey of  $S$ . It contradicts the

fact that  $K$  is a key. Hence we must have  $x \in \Omega^{(0)}$ .

Corollary 1.3.

$$(GR')_S^+ \setminus G \subseteq \Omega^{(0)}.$$

Proof. From direct use of lemma 1.3 with

$$X=G, Y=R'=R \setminus L \subseteq \Omega^{(0)}$$

Example 1.1. We consider one example in which

$$R' \subset (GR')_S^+ \setminus G \subseteq \Omega^{(0)},$$

showing that our lemma is non trivial.

Let  $S = \langle \Omega, F \rangle$  be a relation scheme where

$$\Omega = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9$$

$$F = \{137 \rightarrow 27, 27 \rightarrow 134, 1238 \rightarrow 489, 1458 \rightarrow 236, \\ 368 \rightarrow 159\}$$

We have:

$$L = \bigcup_{i=1}^6 L_i = 12345678; R = \bigcup_{i=1}^6 R_i = 123456789$$

$$R' = R \setminus L = 9; G = \Omega \setminus \left( \bigcup_{i=1}^6 (R_i \setminus L_i) \right) = \Omega \setminus 1234569 = 78.$$

$$(GR')_S^+ = (789)_S^+ = 1234789$$

$$(GR')_S^+ \setminus G = 12349 \supset 9$$

Lemma 1.4. Let  $S = \langle \Omega, F \rangle$  be a relation scheme and  $Z_1, Z_2 \subseteq \Omega$ .

If  $Z_1 \cap Z_2 = \emptyset$ , then

$$(S-Z_1)-Z_2 = S-Z_1Z_2.$$

Proof. Put  $S_1 = S - Z_1 = \langle \Omega_1, F_1 \rangle$ ,

and  $S_2 = S_1 - Z_2 = \langle \Omega_2, F_2 \rangle$ .

Then

$$\Omega_1 = \Omega \setminus Z_1; F_1 = \{L_i \setminus Z_1 \rightarrow R_i \setminus Z_1 \mid i=1, 2, \dots, k\}.$$

Since  $Z_1 \cap Z_2 = \emptyset$ , so  $Z_2 \subseteq \Omega \setminus Z_1 = \Omega_1$ ,

Consequently  $\Omega_2 = \Omega_1 \setminus Z_2 = (\Omega \setminus Z_1) \setminus Z_2 = \Omega \setminus (Z_1 \cup Z_2)$  and

$$F_2 = \{(L_i \setminus Z_1) \setminus Z_2 \rightarrow (R_i \setminus Z_1) \setminus Z_2 \mid i=1, 2, \dots, k\}$$

$$= \{L_i \setminus Z_1 Z_2 \rightarrow R_i \setminus Z_1 Z_2 \mid i=1, 2, \dots, k\}.$$

Corollary 1.4. If  $Z_2 \subseteq \Omega^{(o)}$ , then

$$(S-G) - Z_2 = S - GZ_2.$$

Proof. By direct application of lemma 1.4 in which  $Z_2 \subseteq \Omega^{(o)}$ ,  
 $Z_1 = G$ ,  $Z_2 \cap Z_1 = \emptyset$

Corollary 1.5. If  $Z_2 \subseteq \Omega^{(o)}$  and  $\tilde{S} = (S-G) - Z_2$ , then  $\mathcal{K}_S = G \oplus \mathcal{K}_{\tilde{S}}$

Proof. Let  $S' = S - G$ .

Following the corollary 1.1:  $\mathcal{K}_S = G \oplus \mathcal{K}_{S'}$ . On the other hand:  
 $\tilde{S} = (S-G) - Z_2 = S' - Z_2$ , so again by corollary 1.2, we have  $\mathcal{K}_{\tilde{S}} = \mathcal{K}_{S'}$ .

$$\text{Hence } \mathcal{K}_S = G \oplus \mathcal{K}_{\tilde{S}}$$

We are now in a position to present an algorithm realizing the translation of a relation scheme.

Let be given a relation scheme  $S = \langle \Omega, F \rangle$  where

$$\Omega = \{A_1, A_2, \dots, A_n\}$$

$$F = \{L_i \rightarrow R_i \mid L_i, R_i \subseteq \Omega, i=1, 2, \dots, k\}$$

Using corollary 1.2, we try to find two subset  $Z_1 \subseteq G$  and  $Z_2 \subseteq \Omega^{(o)}$  as great as possible, and then, we define a new relation

scheme as follow:

$$\tilde{S} = S - Z_1 Z_2 .$$

Then  $\mathcal{K}_S = Z_1 \oplus \mathcal{K}_{\tilde{S}}$ .

From the definition of the translation of relation scheme,  $\tilde{S}$  in general is more simple w.r.t.  $S$  in the following points:

- The number of  $FD$  in  $\tilde{S}$  is in general less then the number of  $FD$  in  $S$ .
- The length side (left, right) of eath  $FD$  in  $\tilde{S}$  is possible shorter than the corresponding one in  $S$ .

We choose:

$$Z_1 = G = \Omega \setminus \bigcup_{i=1}^k (R_i \setminus L_i).$$

$$Z_2 = (GR')_S^+ \setminus G \subseteq \Omega^{(0)}$$

(Recall that  $R' = R \setminus L$ ).

Then  $\tilde{S} = S - Z_1 Z_2$ .

Taking account of the expressions of  $Z_1$  and  $Z_2$ , we have

$$Z_1 Z_2 = (GR')_S^+.$$

Moreover, one can remove from  $\tilde{S}$  the  $FD$  of the form:  $\emptyset \rightarrow \emptyset$ ,  $X \rightarrow \emptyset$ ,  $\emptyset \rightarrow X$ , ( $X \neq \emptyset$ ) while preserving  $\mathcal{K}_{\tilde{S}}$ .

Example 1.2. Let be given  $S = \langle \Omega, F \rangle$

where  $\Omega = 123456789$ ;

$$F = \{137 \rightarrow 27, 27 \rightarrow 134, 1238 \rightarrow 489, 1458 \rightarrow 236, \\ 368 \rightarrow 159, 7 \rightarrow 23\}$$

We have  $L = 12345678$ ;  $R = 123456789$ ;

$$R' = R \setminus L = 9; G = 78; Z = (GR')_S^+ = (789)_S^+ = 1234789.$$

Define the new relation scheme:

$$\tilde{S} = S - (GR')_S^+ = \langle \tilde{\Omega}, \tilde{F} \rangle$$

where

$$\tilde{\Omega} = \Omega \setminus Z = 56, \quad Z = (GR)_S^+$$

$$\begin{aligned} F &= \{L_i \setminus Z \rightarrow R_i \setminus Z \mid i=1, 2, \dots, k\} = \\ &= \{\emptyset \rightarrow \emptyset \quad (\text{will be eliminated}); \\ &\quad \emptyset \rightarrow \emptyset \quad (\text{will be eliminated}); \\ &\quad \emptyset \rightarrow \emptyset \quad (\text{will be eliminated}); \\ &\quad 5 \rightarrow 6 \\ &\quad 6 \rightarrow 5 \\ &\quad \emptyset \rightarrow \emptyset \quad (\text{will be eliminated})\} = \\ &= \{5 \rightarrow 6; 6 \rightarrow 5\} \end{aligned}$$

It is easy to see that  $\tilde{S}$  has only two keys which are 5 and 6, i.e.  $\mathcal{K}_{\tilde{S}} = \{5, 6\}$ . Consequently,

$$\mathcal{K}_S = G \oplus \mathcal{K}_{\tilde{S}} = 78 \oplus \{5, 6\} = \{578, 678\}.$$

## §2. THE BALANCED RELATION SCHEME

Definition 2.1. The relation scheme  $S = \langle \Omega, F \rangle$  is called balanced if the following conditions hold:

- (i)  $\bigcup_{i=1}^k L_i = \bigcup_{i=1}^k R_i = \Omega;$
- (ii)  $L_i \cap R_i = \emptyset, \quad \forall i=1, 2, \dots, k;$
- (iii)  $\forall i, j=1, 2, \dots, k; i \neq j \text{ implies } L_i \neq L_j.$

where  $\Omega = \{A_1, A_2, \dots, A_n\}$

$$F = \{L_i \rightarrow R_i \mid L_i, R_i \subseteq \Omega, i=1, 2, \dots, k\}.$$

From the definition 2.1, we can prove the following properties of a balanced relation scheme.

### Proposition 2.1.

Let  $S = \langle \Omega, F \rangle$  be a balanced relation scheme (b.r.s). Then:

1.  $G = \emptyset$ ;
2. If  $|\Omega| \leq 1$  then  $\mathcal{K}_S = \{\emptyset\}$ ;
3.  $\emptyset \notin \mathcal{K}_S \leftrightarrow |\mathcal{K}_S| \geq 2$
4.  $\forall Z \subseteq \Omega$ ,  $S-Z$  is a b.r.s.

Proof.

1. By the definition of a b.r.s, we have:

$$G = \Omega \setminus \bigcup_{i=1}^k (R_i \setminus L_i) = \Omega \setminus \bigcup_{i=1}^k R_i = \Omega \setminus \Omega = \emptyset$$

2. If  $\Omega = \emptyset$ , it is obvious that  $\mathcal{K}_S = \{\emptyset\}$ .

If  $\Omega = \{A\}$ . From (i) (def.2.1) we have  $R = L = \Omega = \{A\}$ .

From (ii),  $F$  contains only two FD:  $\{A\} \rightarrow \emptyset$  and  $\emptyset \rightarrow \{A\}$ , showing that  $\emptyset$  is the unique key of  $\Omega$ .

3. Suppose  $|\mathcal{K}_S| \geq 2$ . Then  $\emptyset \notin \mathcal{K}_S$ , since otherwise  $\emptyset$  will be the unique key of  $S$ .

In the inverse direction, suppose that  $\emptyset \notin \mathcal{K}_S$ . Then  $\mathcal{K}_S$  has at least two elements since otherwise, if  $\mathcal{K}_S = \{K\}$  then from  $G=K$  and  $G=\emptyset$  it follows that  $K=\emptyset$ .

4. This property is obvious.

Theorem 2.1. Let  $S = \langle \Omega, F \rangle$  be an arbitrary given relation scheme, where  $\Omega = \{A_1, \dots, A_n\}$ ,  $F = \{L_i \rightarrow R_i, i=1, 2, \dots, k\}$ . Then there exists a b.r.s.  $S = \langle \Omega, F \rangle$  such that  $\mathcal{K}_S = G \oplus \mathcal{K}_{\tilde{S}}$ , where  $G$  is the intersection of all keys in  $S$ .

Proof. Without loss of, generality, we can always suppose that, for the relation scheme  $S$ ,

$$L_i \cap R_i = \emptyset, \quad \forall i=1, 2, \dots, k.$$

(Otherwise, we replace  $S$  by  $S_1 = \langle \Omega, F_1 \rangle$ , where

$F_1 = \{L_i \rightarrow R_i \setminus L_i \mid (L_i \rightarrow R_i) \in F, i=1, 2, \dots, k\}$ . It is easy to show that  $F^+ = F_1^+$  [7] and therefore  $\mathcal{K}_S = \mathcal{K}_{S_1}$ ).

We construct the b.r.s. as follows:

1. Compute:

$$L = \bigcup_{i=1}^k L_i; R = \bigcup_{i=1}^k R_i; R' = R \setminus L$$

$$G = \Omega \setminus \bigcup_{i=1}^k (R_i \setminus L_i) = \Omega \setminus R;$$

$$Z = (GR')^+$$

Now consider the relation scheme:

$$S' = \langle \Omega', F' \rangle = S - Z,$$

$$\text{where } \Omega' = \Omega \setminus Z,$$

$$F' = \{L_i' \rightarrow R_i' \mid i=1, 2, \dots, k\}$$

$$\text{with } L_i' = L_i \setminus Z, R_i' = R_i \setminus Z.$$

It is obvious that:  $L_i' \cap R_i' = \emptyset, i=1, 2, \dots, k;$

$$V = \bigcup_{i=1}^k L_i' = L \setminus Z \quad \text{and} \quad W = \bigcup_{i=1}^k R_i' = R \setminus Z$$

2. We shall prove that:  $V \subseteq \Omega' \subseteq W \subseteq V$  to deduce that  $V = \Omega' = W.$

Indeed, if  $x \in V$  so  $x \in L$  and  $x \notin Z.$

It is obvious that  $x \in \Omega.$

Consequently  $x \in \Omega \setminus Z = \Omega'. \text{ Hence } V \subseteq \Omega'.$

Now let  $x \in \Omega' = \Omega \setminus Z,$  then  $x \notin Z.$

Since  $x \notin Z = (GR')_S^+ \supseteq GR',$  so  $x \in G$  and  $x \notin R'.$

Re call that  $G = \Omega \setminus \bigcup_{i=1}^k (R_i \setminus L_i),$  we find that  $x \in \bigcup_{i=1}^k (R_i \setminus L_i).$

Hence there exists  $j \in \{1, 2, \dots, k\}$  such that  $x \in R_j,$  showing that  $x \in R.$

From  $x \in R$  and  $x \notin Z,$  we deduce  $x \in R \setminus Z = W. \text{ Therefore } \Omega' \subseteq W.$

Finally if  $x \in W = R \setminus Z$  so  $x \in R$  and  $x \notin Z. \text{ Arguing as above, we have}$

$$x \in G \quad \text{and} \quad x \notin R' = R \setminus L.$$

Since  $x \in R$ , and  $x \notin R \setminus L$ , we deduce  $x \in L$ .

From  $x \in L$  and  $x \notin Z$ , we have  $x \in L \setminus Z$  showing that  $W \subseteq V$ .

Thus we have shown:  $\bigcup_{i=1}^k L_i^+ = \bigcup_{i=1}^k R_i^+ = \Omega^+$ .

3. If there are several FD of  $F'$  with the same left side, we can replace them by a FD which has the left side as the common one, and its right side is the union of their right sides.

It is easy to see that the above transformation does not change the closure of  $F'$  and thus, the set  $S$ , too.

Denote by  $\tilde{S}$ , the relation scheme obtained from  $S'$  after performing the above substitutions. It is clear that  $\tilde{S}$  is the desired balanced relation scheme, and by corollary 1.5

$$\mathcal{K}_S = G \oplus \mathcal{K}_{\tilde{S}}.$$

Definition 2.2. Let  $S = \langle \Omega, F \rangle$  be a b.r.s., where

$$F = \{L_i \rightarrow R_i \mid i=1, 2, \dots, k\}.$$

Denote

$$\mathcal{L}_S = \{L_i \mid i=1, 2, \dots, k\},$$

the set of all left sides of  $F$ .

Construct the directed graph  $\mathcal{G}_S$  as follows:

- (1)  $\mathcal{L}_S$  is the set of nodes of  $\mathcal{G}_S$
- (2)  $(L_i, L_j)$  is an arc of  $\mathcal{G}_S$  iff  $L_i \supset L_j$  and there is no  $L_k$  such that  $L_i \supset L_k \supset L_j$ .

Let  $\mathcal{L}_S^*$  is the set of all terminals nodes of  $\mathcal{G}_S$ , e.i. nodes for which the outdegree is equal to zero. The members of  $\mathcal{L}_S^*$  are called minimal left sides of  $S$ .

Lemma 2.1. Let  $L_i \in \mathcal{L}_S^*$ .

Then

$$L_i \in \mathcal{K}_S \leftrightarrow (L_i)_S^+ = \Omega$$



Proof. The if part is trivial. Suppose that  $L_i \notin \tilde{\mathcal{L}}_S$  and  $L_i^+ = \Omega$ . So  $L_i$  is a superkey of  $S$ . Since  $L_i$  is minimal, then for all  $X \subset L_i$  we have  $X^+ = X \subset L_i$ , showing that  $L_i$  is a key of  $S$ .

Lemma 2.2. Let  $K$  be a key of  $S = \langle \Omega, F \rangle$ . Then  $(Z)_S^+ \cap (K \setminus Z) = \emptyset$  for all  $Z \subseteq K$ .

Proof. Let us denote by  $Y = (Z)_S^+ \cap (K \setminus Z)$ . It is clear that  $Y \subseteq (Z)_S^+$ ,  $Y \subseteq K$  and  $Y \cap Z = \emptyset$ . Therefore we can write  $K = Z | Y | X$  (a partition of  $K$ ) and have (c.f. proposition 1.1)

$$(ZX)_S^+ = ((Z)_S^+ X)^+ = ((Z)_S^+ YX)^+ \supseteq (ZYX)^+ = \Omega.$$

Since  $K$  is a key, so  $ZX = K$  showing that  $Y = \emptyset$ .

The following corollary is immediate:

Corollary 2.1. If  $K \in \mathcal{K}_S$  and  $Z \subseteq K$ , then:

- 1)  $(Z)_S^+ \cap K = Z$
- 2)  $K \setminus (Z)_S^+ = K \setminus Z$ .

Lemma 2.3. Let  $L_i$  be an arbitrary element of  $\mathcal{L}_S$ , and  $\tilde{S} = S - (L_i)_S^+$ .

Then the elements of  $L_i \oplus \mathcal{K}_{\tilde{S}}$  are superkeys of  $S$ .

Proof. Let  $Z = (L_i)_S^+$ . Then  $\forall \tilde{K} \in \mathcal{K}_{\tilde{S}}$ .  $(L_i \tilde{K})_S^+ = ((L_i)_S^+ \tilde{K})_S^+ = (Z \tilde{K})_S^+ = Z(\tilde{K})_{\tilde{S}}^+ = Z\tilde{Z} = \Omega$  by virtue of proposition 1.1 and theorem 1.1, showing that  $L_i \tilde{K}$  is a superkey of  $S$ .

Theorem 2.2. (key representation)

Let  $S = \langle \Omega, F \rangle$  be a b.r.s.

Then each key of  $S$  can be represented in the form:

$$K = L_i \tilde{K}$$

where  $L_i$  is a minimal left side of  $S$ , i.e.  $L_i \notin \tilde{\mathcal{L}}_S$ , and  $\tilde{K}$  is a

key of the b.r.s.  $\tilde{S} = S - (L_i)_S^+$ .

Proof. Let  $K$  be a key of  $S$ , i.e.  $K \in \mathcal{K}_S$ .

If  $K = \Omega$  then, of course,  $K$  contains all elements of  $S$ .

If  $K \neq \Omega$ , so  $K \subset (K)_S^+ = \Omega$ . That means, there exist  $L_j \in \mathcal{L}_S$ , such that  $L_j \subseteq K$  and  $R_j \setminus K \neq \emptyset$ .

(This follows from the algorithm to find the closure of a set of attributes w.r.t.  $F$ ).

Starting from the node  $L_j$  of the graph  $\mathcal{G}_S$ , we move along the arcs until a node  $L_i \in \mathcal{L}_S$  is reached. Obviously  $L_i \subseteq L_j$ . Thus we have proved that:

$$\forall K \in \mathcal{K}_S, \exists L_i \in \mathcal{L}_S \text{ such that } L_i \subseteq K.$$

Put  $Z = (L_i)_S^+$ .

If  $Z = \Omega$ , so, by lemma 2.1,  $L_i$  is a key of  $S$  and we have  $K = L_i \cup \emptyset$ .

In this case  $\tilde{S} = S - Z = S - \Omega = \langle \emptyset, \{\emptyset \rightarrow \emptyset\} \rangle$ , and clearly  $\emptyset$  is a key of  $\tilde{S}$ .

If  $Z \subset \Omega$ , we can write

$$K = L_i \cup \tilde{K} \quad \text{i.e.} \quad K = L_i \cup \tilde{K}, \quad L_i \cap \tilde{K} = \emptyset.$$

We shall prove that  $\tilde{K}$  is a key of  $\tilde{S}$ .

By lemma 2.2, we have:

$$(L_i)_S^+ \cap (K \setminus L_i) = Z \cap \tilde{K} = \emptyset, \quad (\tilde{K} = K \setminus L_i)$$

Consequently  $\tilde{K} \subseteq \bar{Z} = \tilde{\Omega}$ .

Moreover  $\tilde{K} = K \setminus L_i = (K \setminus (L_i)_S^+)$  (corollary 2.1). Therefore, by lemma 1.1,  $\tilde{K}$  is a superkey of  $\tilde{S}$ .

Now, suppose that there is  $\tilde{K}' \subset \tilde{K}$  and  $\tilde{K}'$  is a key of  $\tilde{S}$ . Again, by lemma 1.1,  $Z \tilde{K}' = (L_i)_S^+ \tilde{K}'$  is a superkey of  $S$ . Thus,

$$\Omega = (Z \tilde{K}')_S^+ = ((L_i)_S^+ \tilde{K}')_S^+ = (L_i \tilde{K}')_S^+$$

showing  $L_i \tilde{K}'$  is a superkey of  $S$ . On the otherhand it is clear

that  $L_i \tilde{K}' \subset K$ .

This contradicts the fact that  $K$  is a key of  $S$ . Hence  $\tilde{K}' = \tilde{K}$ , i.e.  $\tilde{K}$  is a key of  $S$ . The theorem 2.2 is completely proved.

Remark 2.1. In general, the inverse of theorem 2.2 is not true. It is quite possible that there exist  $L_i \notin \tilde{\mathcal{L}}_S$  while  $L_i$  does not contained in any key of  $S$ .

Example 2.2.

$$\Omega = 12345; F = \{24 \rightarrow 35, 15 \rightarrow 4, 53 \rightarrow 124, 25 \rightarrow 134\}$$

We have

$$\mathcal{L}_S = \tilde{\mathcal{L}}_S = \{24, 15, 53, 25\}$$

The graph  $\mathcal{G}_S$  consists of all disjoint nodes, (Fig.1.)

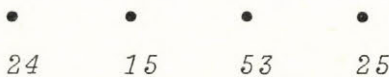


Fig.1.

Direct computation, shows that:

$$(24)_S^+ = (53)_S^+ = (25)_S^+ = \Omega$$

Therefore 24, 53 and 25 are keys of  $S$ .

On the other hand:

$$(15)_S^+ = 154 \neq \Omega$$

It is clear that 15 is not contained in any key of  $S$  because, 152 contains the key 25 and 153 contains the key 35.

Corollary 2.2.

Let  $L_i \notin \tilde{\mathcal{L}}_S$ ;  $\tilde{S} = S - (L_i)_S^+$ .

If  $K \in L_i \oplus \mathcal{K}_{\tilde{S}}$  and except  $L_i$ ,  $K$  does not contain any other minimal left side  $L_j \in \tilde{\mathcal{L}}_S$ , then  $K$  is a key of  $S$ .

Proof. Following lemma 2.3,  $K$  is a superkey of  $S$ . Suppose that  $K' \subsetneq K$  and  $K'$  is a key of  $S$ .

We shall prove that  $K' = K$ .

Indeed, since  $K$  contains only  $L_i$ , so  $K'$  at most contains only  $L_i$ .

If  $K'$  does not contain  $L_i$ , then  $(K')_S^+ = K' \neq \Omega$ . Thus  $K'$  must contain  $L_i$ . We have  $K' = L_i \tilde{K}'$ . Since  $K \in L_i \oplus \mathcal{K}_S$ ,  $K = L_i \tilde{K}$ ,  $\tilde{K}' \in \mathcal{K}_{\tilde{S}}$ . By the proof of theorem 2.2 (keyrepresentation). We have  $\tilde{K}' \in \mathcal{K}_{\tilde{S}}$ .

From  $L_i \tilde{K}' = K' \subsetneq K = L_i \tilde{K}$  and  $L_i \cap \tilde{K}' = L_i \cap \tilde{K} = \emptyset$  we deduce  $\tilde{K}' \subsetneq \tilde{K}$ .

Since  $\tilde{K}'$  and  $\tilde{K}$  are keys of  $\tilde{S}$ , so  $\tilde{K}' = \tilde{K}$ . Thus  $K' = K$ .

SUMMARY

We are now ready to present a general scheme to transform an arbitrary relation scheme into a balanced relation scheme and to find all its keys.

Let be given a relation scheme

$$S = \langle \Omega, F \rangle$$

where

$$\Omega = \{A_1, A_2, \dots, A_n\}$$

$$F = \{L_i \rightarrow R_i \mid L_i, R_i \subseteq \Omega; i=1, 2, \dots, k\}$$

$$L_i \cap R_i = \emptyset, i=1, 2, \dots, k.$$

Step 1. Compute  $L = \bigcup_{i=1}^k L_i$ ;  $R = \bigcup_{i=1}^k R_i$ ;

$$R' = R \setminus L; G = \Omega \setminus \bigcup_{i=1}^k (R_i \setminus L_i) = \Omega \setminus R.$$

$$Z = (GR')_S^+$$

Step 2. Define  $S' = \langle \Omega', F' \rangle = S - Z$

where

$$\Omega' = \Omega \setminus Z;$$

$$F' = \{L_i \setminus Z \rightarrow R_i \setminus Z \mid i=1, 2, \dots, k\}$$

Eliminate from  $F'$  functional dependencies of the form:

$$\emptyset \rightarrow \emptyset, \emptyset \rightarrow X, X \rightarrow \emptyset \quad (X \neq \emptyset)$$

Perform the grouping operation for  $FD$  of  $F'$  which have the same left side.

Thus, we obtain, the b.r.s.  $\tilde{S} = \langle \tilde{\Omega}, \tilde{F} \rangle$

Step 3. Find all keys of  $\tilde{S}$ .

Construct:  $\mathcal{L}_{\tilde{S}}$  - the set of all left sides of  $\tilde{F}$ ; the graph  $\mathcal{G}_{\tilde{S}}$ ;

$\tilde{\mathcal{L}}_{\tilde{S}}$  - the set of all minimal left sides of  $\tilde{F}$ .

Let

$$\tilde{\mathcal{L}}_{\tilde{S}} = \{L_1, L_2, \dots, L_l\}$$

Compute  $Z_i = (L_i)_S^+$ ,  $i=1, 2, \dots, l$ .

If  $Z_i = \tilde{\Omega}$  then  $L_i \in \mathcal{K}_{\tilde{S}}$ .

Denote by  $I = \{j \mid Z_j \neq \tilde{\Omega}, j \in \{1, 2, \dots, l\}\}$

For  $\tilde{Z}_j \neq \tilde{\Omega}$ , consider the b.r.s.

$$\mathcal{Y}_j = \tilde{S} - Z_j \quad \forall j \in I$$

Repeat the step 3 for the relation schemes  $\mathcal{Y}_j$ . Suppose that at some moment we found all keys of  $\mathcal{Y}_j$ ,  $j \in I$

$$\mathcal{K}_j = \{K_1^{(j)}, K_2^{(j)}, \dots, K_{s_j}^{(j)}\}$$

To complete the set  $\mathcal{K}_{\tilde{S}}$ , we perform as follows:

Consider sequentially the sets  $L_{j,K_t^{(j)}}$ , for each  $j \in I$ ,  
 $t=1, 2, \dots, s_j$

- If  $L_{j,K_t^{(j)}}$  contains in a key ready found of  $\mathcal{K}_{\tilde{S}}$ , so we omit it;

- If  $L_{j,K_t^{(j)}}$  contains no element of  $\mathcal{L}_{\tilde{S}}$  but (except)  $L_j$  then  $L_{j,K_t^{(j)}} \in \mathcal{K}_{\tilde{S}}$ .

- otherwise, use algorithm 3 in [ ] to check whether  $L_{j,K_t^{(j)}}$  is a key.

Step 4. Compute  $\mathcal{K}_S = G \oplus \mathcal{K}_{\tilde{S}}$

Example 2.2 Let be given  $S = \langle \Omega, F \rangle$

where

$$\Omega = 1234567$$

$$F = \{167 \rightarrow 3,$$

$$34 \rightarrow 5,$$

$$39 \rightarrow 4,$$

$$3 \rightarrow 2,$$

$$157 \rightarrow 6,$$

$$137 \rightarrow 5\}$$

Step 1.

$$L = \bigcup_{i=1}^6 L_i = 135674;$$

$$R = \bigcup_{i=1}^6 R_i = 23456;$$

$$R' = R \setminus L = 2$$

$$G = \Omega \setminus R = 1234567 \setminus 34526 = 17$$

$$Z = (GR^3)_S^+ = (172)_S^+ = 172$$

Step 2.

$$S' = \langle \Omega', F' \rangle$$

$$\Omega' = \Omega \setminus Z = 3456$$

$$F' = \{ 6 \rightarrow 3,$$

$$34 \rightarrow 5,$$

$$35 \rightarrow 4,$$

$$3 \rightarrow \emptyset,$$

(will be eliminated)

$$5 \rightarrow 6,$$

$$3 \rightarrow 5, \}$$

Thus  $S = \langle \tilde{\Omega}, \tilde{F} \rangle$  is a b.r.s. where  $\tilde{\Omega} = 3456$

$$F = \{ 6 \rightarrow 3,$$

$$34 \rightarrow 5,$$

$$35 \rightarrow 4,$$

$$5 \rightarrow 6,$$

$$3 \rightarrow 5, \}$$

Step 3. Find all keys of  $\tilde{S}$

$$\mathcal{L}_{\tilde{S}} = \{ 6, 34, 35, 5, 3 \}$$

The graph  $\mathcal{G}_{\tilde{S}}$  as in Fig.2.

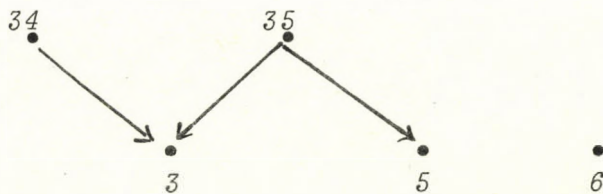


Fig.2.

We have:

$$(3)_S^+ = 3546 = \tilde{\Omega}, \text{ i.e. } 3 \in \mathcal{K}_{\tilde{S}}$$

$$(5) \overset{+}{\mathcal{K}}_{\tilde{S}} = 5634 = \tilde{\Omega}, \text{ i.e. } 5 \in \mathcal{K}_{\tilde{S}}$$

$$(6) \overset{+}{\mathcal{K}}_{\tilde{S}} = 6354 = \tilde{\Omega}, \text{ i.e. } 6 \in \mathcal{K}_{\tilde{S}}.$$

Thus  $\mathcal{K}_{\tilde{S}} = \{3, 5, 6\}$

Step 4.

$$\mathcal{K}_S = G \oplus \mathcal{K}_{\tilde{S}} = 17 \oplus \{3, 5, 6\} = \{163, 175, 176\}$$

The given relation scheme has three keys: 173, 175 and 176.

We close our paper with some open problems.

### §3. SOME OPEN PROBLEMS

1. Description of the greatest as possible subset of  $\Omega^{(0)}$ .
2. Find the necessary and sufficient condition for which a minimal left side is strictly contained in one key.
3. Find the sufficient condition for which all minimal left sides of the  $FD$  of a balanced relation scheme are keys.
4. Transform the above general scheme (§2, Summary) into an efficient computer algorithm.

### ACKNOWLEDGEMENT

The authors wish to thank Dr.A. Bekessy for his valuable remarks and suggestions.



## REFERENCES

- [1] Codd, E.F., A relational model of data for large shared data banks.  
Commun. ACM, vol.13, June 1970.
- [2] Armstrong W.W., Dependency structure of database relationships.  
Information Processing 74, North Holland pub.Co.Amsterdam, 1974
- [3] Demetrovics J. and Gyepesi Gy., Some generalized type functional dependencies formalized as equality set on matrices. Discrete Applied Mathematics 6 (1983), North-Holland Publishing Company.
- [4] Ho Thuan and Le van Bao, Translation of relational schemes, Közlemények 30/19.  
MTA-SzTAKI, Budapest
- [5] Bekessy A. and Demetrovics J., Contribution to the theory of Database relations,  
Discrete Mathematics, 27 (1970).
- [6] Ho Thuan and Le van Bao, Some results about keys of relational schemas,  
Acta Cybernetica, Tomus 7, Fasciculus 1, 1985, Szeged, Hungary
- [7] Ullman, J., Principles of database systems,  
Computer Science Press, Second edition, 1982.
- [8] Beeri, C. and Vardi, M.Y., Formal systems for tuple and equality generating dependencies.  
SIAM J. Comput. 13, 1 (Feb. 1984).

A kulcs-reprezentáció problémája és a kiegyensúlyozott reláció  
séma

J. DEMETROVICS, HO THUAN, NGUYEN XUAN HUY, LE VAN BAO

Összefoglaló

A szerzők bevezetik az u.n. kiegyensúlyozott reláció séma ill. a reláció séma eltolás-operátorának fogalmát. E fogalmak segítségével megadják egy reláció séma összes kulcsának reprezentációját.

Проблема представления ключей и уравновешенная реляционная  
схема

Я. Деметрович, Хо Туан, Нгуен Хуан Хуи, Ле Ван Бао

Резюме

В статье вводятся понятия уравновешенной реляционной схемы и оператора трансляций реляционной схемы. С помощью этих понятий дается представление всех ключей любой реляционной схемы.

О СЛОЖНОСТИ РЕАЛИЗАЦИИ ОДНОЙ ТРЕУГОЛЬНОЙ МАТРИЦЫ  
 ВЕНТИЛЬНЫМИ СХЕМАМИ \* РАЗНОЙ ГЛУБИНЫ

Гал Анна

Введем следующие обозначения:

- [a] - наименьшее целое число, не меньше числа a
- [a] - наибольшее целое число, не больше числа a
- $t_{ij}$  - элемент матрицы стоящий в i-той строке и j-том столбце
- $T_n$  - матрица с n строками и n столбцами, для которой

$$t_{ij} = \begin{cases} 0, & \text{если } i \geq j \\ 1, & \text{если } i < j \end{cases}$$

$L(S)$  - число вентилях в схеме S

$$L^{(r)}(n) = L^{(r)}(T_n)$$

$L^{(r)}(T_n) = \min L(S)$  по всем схемам S глубины  $\leq r$ , реализующим  $T_n$ .

Очевидно, что  $L^{(1)}(T_n) \sim \frac{n^2}{2}$  при  $n \rightarrow \infty$ .

В [2] доказано, что  $L^{(2)}(T_n) \sim n \cdot \log n$  при  $n \rightarrow \infty^{**}$ .

В настоящей работе получены верхние оценки для схем глубины  $r \geq 3$ :

$$L^{(r)}(T_n) \leq c \cdot n \cdot K_r(n) \tag{1}$$

где c некоторая константа, а  $K_r(n)$  последовательность функций

\* Определение вентилях схем см. в [1].

\*\* Здесь и ниже  $\log$  означает логарифм по основанию 2, а асимптотические соотношения рассмотрим при  $n \rightarrow \infty$ .

определенная следующим образом:  $K_1(n) = \sqrt{n}$ ,  $K_2(n) = \log n$ ,

$$K_r(n) = \min\{i \mid \overbrace{K_{r-2}(\dots(K_{r-2}(n))\dots)}^i \leq 2\}.$$

Заметим, что  $K_3(n) = \lceil \log \log n \rceil$ ,  $K_4(n) = \log^* n$ , а уже  $K_6(n)$  является очень медленно возрастающей функцией. Отметим, связь функций  $K_r(n)$  с функцией обратной к функции Аккерманна /определение см. в [3] стр. 72/. Для  $\forall r = 2s$ ,  $s \geq 1$  справедливо следующее соотношение:

$$K_r(n) = \min\{i \mid A(2, i, s+2) \geq n\},$$

где  $A$  есть функция Аккерманна.

Для доказательства утверждения (1) для любой матрицы  $T_n$  построим схему  $S$  глубины  $r$  реализующую  $T_n$ , сложности  $L(S) \leq c \cdot n \cdot K_r(n)$ . Строкам матрицы  $T_n$  ставим в соответствие входы, столбцам выходы схемы  $S$ . Перенумеруем входы и выходы от 1 до  $n$ . Схема  $S$  будет состоять из  $K$  подсхем. Обозначим их через  $S^{(1)}, S^{(2)}, \dots, S^{(K)}$ .

Построение схемы  $S^{(1)}$ :

Разобьем строки и столбцы матрицы  $T_n$  соответственно на  $k_1$  групп, в каждой из которых не больше чем  $\lceil \frac{n}{k_1} \rceil$  строк /столбцов/.

Перенумеруем группы строк и группы столбцов от 1 до  $k_1$ . Возьмем минимальную схему  $S_{k_1}$  глубины  $r-2$ , реализующую матрицу  $T_{k_1}$ . Тогда  $L(S_{k_1}) = L^{(r-2)}(k_1)$ . /Заметим, что в силу наших обозначений  $T_{k_1}$  имеет ту же структуру как  $T_n$ , только число ее строк и столбцов равно  $k_1$ . / Перенумеруем входы и выходы схемы  $S_{k_1}$  от 1 до  $k_1$ . Подставим  $S_{k_1}$  в схему  $S$  следующим образом: входы схемы  $S$  соответствующие строкам матрицы  $T_n$   $i$ -той группы соединим  $i$ -тым входом схемы  $S_{k_1}$ , а выходы схемы  $S$  соответствующие столбцам  $i$ -той группы соединим  $i$ -тым выходом схе-

мы  $S_{k_1}$ . Схему /часть схемы  $S$ / которую так получим обозначим через  $S^{(1)}$ . Видно, что  $L(S^{(1)}) = 2n + L^{(r-2)}(k_1)$ .

Построение схемы  $S^{(j+1)}$ :

$S^{(j+1)}$  будет состоять из  $k_1 \cdot k_2 \dots k_j$  частей, каждая из которых реализует подматрицу из элементов стоящих на пересечениях строк и столбцов одной группы /номер группы строк и группы столбцов то же самое/. Тут имеются в виду группы строк и столбцов полученные при построении схемы  $S^{(j)}$ . Каждая подматрица имеет ту же структуру как исходная матрица  $T_n$ , и размер  $\leq \lceil \frac{n}{k_1 \cdot k_2 \dots k_j} \rceil$ . При построении  $S^{(j+1)}$  строки и столбцы каждой подматрицы разобьем на  $k_{j+1}$  групп, и для каждой подматрицы поступим аналогично построению схемы  $S^{(1)}$ . Будет выполняться:

$$L(S^{(j+1)}) = 2n + k_1 \cdot k_2 \dots k_j \cdot L^{(r-2)}(k_{j+1}).$$

Пусть  $K$  первое число, для которого выполнено:

$$\lceil \frac{n}{k_1 \cdot k_2 \dots k_{K-1}} \rceil \leq 2.$$

Тогда при построении схемы  $S^{(K)}$  будем иметь дело с подматрицами число строк и столбцов которых  $\leq 2$ . Очевидно, что

$$L(S^{(K)}) \leq k_1 \cdot k_2 \dots k_{K-1}. \quad \text{Так полученная схема } S = \bigcup_{j=1}^K S^{(j)}$$

реализует  $T_n$ , и сложность ее

$$L(S) \leq K \cdot 2n + \sum_{j=1}^{K-1} k_1 \dots k_{j-1} L^{(r-2)}(k_j) + k_1 \dots k_{K-1} \quad (2)$$

$k_j$ -тые выбираем следующим образом:

$$k_1 = \lceil \frac{n}{K_{r-2}(n)} \rceil$$

$$k_j = \lceil \frac{\overbrace{K_{r-2}(\dots(K_{r-2}(n))\dots)}^{j-1 \text{ раз}}}{\underbrace{K_{r-2}(\dots(K_{r-2}(n))\dots)}_j \text{ раз}} \rceil \quad \text{для } j=2, \dots, K-1$$

Очевидно, что  $k_1 \dots k_{K-1} < n$  выполняется для  $\forall r \geq 3$ .

Пусть  $r = 3$ . Тогда  $L^{(r-2)}(k_j) \leq \frac{1}{2} k_j^2$ , и по определению  $k_j$ -тых для  $\forall j=1, \dots, K-1$ :  $k_1 \dots k_{j-1} \cdot k_j^2 \leq n$ .

Пусть  $r > 3$ . Так как  $L^{(2)}(n) \sim n \cdot \log n$ , можем использовать, что  $L^{(r-2)}(k_j) \leq c \cdot k_j \cdot K_{r-2}(k_j)$ . По определению  $k_j$ -тых

$$k_1 \dots k_{j-1} \cdot k_j \cdot K_{r-2}(k_j) \leq n \cdot \underbrace{\frac{K_{r-2}(k_j)}{K_{r-2}(\dots(K_{r-2}(n))\dots)}}_{j \text{ раз}} \leq n,$$

так как функции  $K_r(n)$  монотонно возрастают. Используя эти неравенства из (2) получим  $L(S) \leq c \cdot n \cdot K$ . Очевидно, что  $K \ll K_r(n)$ . Утверждение (1) доказано.

В заключении заметим, что из доказательства утверждения видно:

$$L^{(3)}(T_n) \lesssim \frac{5}{2} n \cdot \log \log n,$$

$$L^{(4)}(T_n) \lesssim 3 n \cdot \log^* n.$$

#### Л и т е р а т у р а:

- [1] Лупанов О.Б.: О вентильных схемах, Acta Cybernetica Tom. 4, Fasc. 4, Szeged, 1980, стр. 311-315.
- [2] Tarján T.G.: Complexity of lattice-configurations, Studia Sci. Math. Hungar. v.10, 1975, pp. 203-211.
- [3] Giorgio Ausiello: Algoritmusok és rekurziv függvények bonyolultságelmélete, Műszaki Könyvkiadó, Budapest, 1984.

Egy háromszögmátrix bonyolultsága különböző mélységű kapuhálózatokkal való realizálása esetén

GÁL Anna

Összefoglaló

$L^{(r)}(T_n)$  jelöli a  $T_n$  mátrix bonyolultságát  $r$ -nél nem nagyobb mélységű kapuhálózatokkal való realizálása esetén. A cikk  $r \geq 3$ -ra a következő eredményt bizonyítja:

$$L^{(r)}(T_n) \leq c \cdot n \cdot K_r(n),$$

ahol  $c$  konstans,  $K_1(n) = \sqrt{n}$ ,  $K_2(n) = \log n$ ,

$$K_r(n) = \min \{i \mid \underbrace{K_{r-2}(\dots(K_{r-2}(n))\dots)}_{i\text{-szer}} \leq 2\}.$$

$i$ -szer

On the complexity of realization of a triangle-matrice by gate circuits of different depths

A. GÁL

Summary

$L^{(r)}(T_n)$  characterise the complexity of matrice  $T_n$ , when the dept of realisation  $\leq r$ . It is shown that for  $r \geq 3$

$$L^{(r)}(T_n) \leq c \cdot n \cdot K_r(n), \text{ where } c \text{ is a constant,}$$

$$K_1(n) = \sqrt{n}, \quad K_2(n) = \log n,$$

$$K_r(n) = \min \{i \mid \underbrace{K_{r-2}(\dots(K_{r-2}(n))\dots)}_{i \text{ times}} \leq 2\}.$$

$i$  times





NOTES ON TRACE LANGUAGES,  
PROJECTION AND SYNTHESIZED COMPUTATION SYSTEMS

*DANG VAN HUNG*

Computer and Automation Institute,  
Hungarian Academy of Sciences

ABSTRACT

Connection between trace languages and projective products is pointed out, the parallel product is defined. The way how it can be used in analyzing synthesized computation systems is presented. Relation of the trace languages to safe Petri-nets is considered, too.

1. INTRODUCTION

Trace languages and projective products are used to represent behaviour of parallel computation systems. Their relation to Petri-nets have been studied by A.Mazukiewicz [8] and E.Knuth [1]. The application of trace languages in representing projective products is presented in [2].

It is worth pointing out the connection between that concepts, how the projective products can be used to represent trace languages. In this paper we shall be concerned with those problems and their applications in studying behaviour of systems synthesized from component systems, which has attracted a great deal of attention to our knowledge.

The next section is devoted to considering the connection between trace languages and projective products. In the third section we attempt to apply this connection to study the

behaviour of synthesized computation systems: The way of verification of such systems is discussed in the last one.

2. REPRESENTING TRACE LANGUAGES VIA PROJECTIVE PRODUCTS

The main objects concerned to in this section are trace languages and projective products. For their details we refer to [1,2,8]. Here we recall only the basic definitions.

Let  $\Sigma$  be a finite alphabet. A binary symmetric and irreflexive relation  $I$  over  $\Sigma$  is said to be an "independency" one. Now define " $\sim$ " as the least equivalence relation on  $\Sigma^*$  satisfying the condition:

$$a, b \in I \Rightarrow w'abw'' \sim w'baw''$$

for all strings  $w', w'' \in \Sigma^*$  and all symbols  $a, b \in \Sigma$ . Traces (with respect to  $I$ ) are defined as equivalence classes of the relation  $\sim$ . The trace containing the word  $w$  (with respect to  $I$ ) is denoted by  $[w]_I$ , and the set of words belonging to trace  $T$  is denoted by  $\{T\}$ .

Suppose that  $w_1, w_2, \dots, w_m \in \Sigma^*$ . The projective product of  $w_1, w_2, \dots, w_m$  (denoted by  $\bigotimes_{i=1}^m w_i$ ) is the set  $\{w \in \Sigma^* \mid \forall i=1, 2, \dots, m, w|_{w_i} = w_i\}$ , where  $w|_v$  denotes the projection of  $w$  into the set of symbols constituting  $v$ .

The constructing of the independency relation, with respect to which the given projective products is a trace, has been presented in [2]. Now we consider the reverse problem. In doing so, we need the following concepts, which has been presented detailly in [6].

Every independency relation is called sir-relation. Let  $I$  be sir-relation, families of subsets  $\overline{\text{ken}}(I)$  and  $\text{ken}(I)$  of  $\Sigma$  are defined as follows:

$$\text{ken}(I) = \{A \mid \forall a, b \in A, (a, b) \in I \cup \text{id} \& \forall c \notin A, \exists a \in A, (a, c) \notin I\},$$

$$\overline{\text{ken}}(I) = \{A \mid \forall a, b \in A, (a, b) \notin I \& \forall c \notin A, \exists a \in A, (a, c) \in I\},$$

where id denotes identify relation.

$\text{ken}(I)$ ,  $\overline{\text{ken}}(I)$  are coverings of  $\Sigma$ . Reversely, from any covering  $\mathcal{A}$  of  $\Sigma$ , we construct a sir-relation  $\text{sir}(\mathcal{A})$  as the following:

$$\text{sir}(\mathcal{A}) = \{(a, b) \mid a \neq b \& \forall A \in \mathcal{A}, a \notin A \text{ or } b \notin A\}.$$

Corollary 1:

For every covering  $\mathcal{A}$  of  $\Sigma$

a/  $\forall B \in \mathcal{A}, \exists A \in \overline{\text{ken}}(\text{sir}(\mathcal{A}))$  such that  $B \subseteq A$ ,

b/  $\text{Sir}(\mathcal{A}) = \text{sir}(\overline{\text{ken}}(\text{sir}(\mathcal{A})))$ .

Now, let  $T$  be a trace language over  $\Sigma$  under  $I$ ,  $T = [L]_I$  ( $L$  is a word-language),  $\mathcal{A}$  be any covering of  $\Sigma$  such that  $\text{sir}(\mathcal{A}) = I$ ,  $\mathcal{A} = \{A_1, A_2, \dots, A_n\}$ . Denote by  $h_i, i=1, 2, \dots, n$  the projections from  $\Sigma$  to  $A_i, i=1, 2, \dots, n$ ;

$$h_i(a) = \text{if } a \in A_i \text{ then } a \text{ else } e,$$

where  $e$  is the empty word. For every  $i=1, \dots, n$ ,  $h_i$  can be extended to a homomorphism from  $\Sigma^*$  to  $A_i^*$  by the usual way.

Theorem 1:  $\forall t \in T$ , there exist uniquely  $w_1, w_2, \dots, w_n, w_i \in A_i^*, i=1, 2, \dots, n$  such that

$$t = \bigotimes_{i=1}^n w_i.$$

Proof: Take  $w \in t$  and put  $w_i = h_i(w), i=1, 2, \dots, n$ . Let  $w' \sim w$ . By definition of relation  $\sim$ , there exist  $w^{(1)}, w^{(2)}, \dots, w^{(m)}$  such that  $w^{(1)} = w, w^{(m)} = w'$  and  $\forall j=1, 2, \dots, m-1$

$$\begin{aligned} w^{(j)} &= w_1^{(j)} a b w_2^{(j)} \\ w^{(j+1)} &= w_1^{(j)} b a w_2^{(j)}, \quad (a, b) \in I. \end{aligned}$$

We show by induction on  $j$  that:

$$\forall i=1,2,\dots,n, \quad h_i(w) = h_i(w^{(j)}).$$

Because  $w^{(1)} = w$ , the case of  $j=1$  is trivial. If  $(a,b) \in I = \text{sir}(\mathcal{A})$  then  $\forall i=1,2,\dots,n, \quad a \notin A_i$  or  $b \notin A_i$ . Hence  $h_i(ab) = h_i(ba)$  and  $h_i(w^{(j)}) = h_i(w^{(j+1)})$ .

The inductive hypothesis gives  $h_i(w^{(j)}) = h_i(w)$ . So  $h_i(w') = h_i(w)$ ,  $\forall i = 1,2,\dots,n, \quad \forall w' \sim w$ . This implies that  $w_1, w_2, \dots, w_n$  are defined and by the definition of projective product,

$$w' \in \bigotimes_{i=1}^n w_i.$$

We have shown that  $\forall t, \forall w \in t, \quad t \subseteq \bigotimes_{i=1}^n h_i(w)$ .

Now, by induction on the length  $|w|$  of  $w$ , we show that

$$\bigotimes_{i=1}^n h_i(w) \subseteq t \quad \forall t \in T, \quad t = [w]_I.$$

When  $|w|=1$ , it is obvious since  $t$  and  $\bigotimes_{i=1}^n h_i(w)$  contains exactly one element.

Suppose that  $\bigotimes_{i=1}^n h_i(w) \subseteq t, \quad \forall w, |w| \leq k, \quad k \geq 1$ . Let  $w'' = wa$  and  $w' \in \bigotimes_{i=1}^n h_i(w'')$ . Then

$$h_i(w'') = \begin{cases} h_i(w) & \text{if } a \notin A_i, \\ h_i(w)a & \text{if } a \in A_i. \end{cases}$$

Because  $a$  has an occurrence in  $w'$ ,  $w'$  can be written in the form

$$w' = y_1 a y_2,$$

where  $y_2$  does not contain an occurrence of letters in  $A_i$  containing  $a$  by  $\text{sir}(\mathcal{A}) = I$ . Hence:

$$h_i(w') = h_i(y_1 a y_2) = \begin{cases} h_i(y_1 y_2), & a \notin A_i \\ h_i(y_1 a), & a \in A_i \end{cases} .$$

Of course,  $h_i(y_1 y_2) = h_i(w)$ ,  $h_i(y_1 a) = h_i(w) a$ . Therefore  $h_i(y_1 y_2) = h_i(w)$ ,  $\forall i=1, 2, \dots, n$  and since that,  $y_1 y_2 \in [w]_I$  by inductive hypotheses. For every  $b$  having an occurrence in  $y_2$ ,  $(a, b) \in I = \text{sir}(\mathcal{A})$ . This implies that  $y_1 a y_2 \sim y_1 y_2 a \sim w a$ .

To complete the proof of theorem 1, we show that the representation  $t = \bigotimes_{i=1}^n w_i$  is unique. But this fact is obvious by definition of projective product.

Combining theorem 1 and theorem 5 (in [2]) gives that a set of words in  $\Sigma^*$  is a trace if and only if it is a projective product of some words.

For the given independency relation  $I$ , we prefer to use this representation in the case of  $\mathcal{A} = \overline{\text{ken}}(I)$  and for the given trace language we prefer to consider the case when  $I$  is the smallest relation, with respect to which  $T$  is trace language.

From the representation of traces, we can define the parallel concatenation of trace languages, which is useful for researching the concurrency of combination of parallel computation systems.

Let  $\Sigma_1, \Sigma_2, \dots, \Sigma_m$  be alphabets (not necessarily disjoint),  $I_1, I_2, \dots, I_m$  be sir-relations on  $\Sigma_1, \Sigma_2, \dots, \Sigma_m$  respectively. Suppose that

$$\overline{\text{ken}}(I_i) = \{A_{n_{i-1}+1}, A_{n_{i-1}+2}, \dots, A_{n_i}\}, \quad i=1, 2, \dots, m,$$

$n_0 = 0$ .

Let  $t_1, t_2, \dots, t_m$  be traces on  $\Sigma_1, \dots, \Sigma_m$  with respect to  $I_1, I_2, \dots, I_m$  respectively. By theorem 1, there exist

$w_1, w_2, \dots, w_{n_m}$  such that:

$$t_i = \bigotimes_{j=n_{i-1}+1}^{n_i} w_j, \quad i=1, 2, \dots, m, \quad w_j \in A_j^*, \quad j=1, 2, \dots, n_m.$$

It follows from theorem 1 that if

$$\bigotimes_{j=1}^{n_m} w_j \neq \emptyset, \quad t = \bigotimes_{j=1}^{n_m} w_j$$

is a trace over  $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \dots \cup \Sigma_m$  (with respect to  $\text{sir}(\bigcup_{i=1}^m \overline{\text{ken}(I_i)})$ ).

Definition 1: The trace  $t$  defined as above is called parallel concatenation of  $t_1, t_2, \dots, t_m$  and is denoted by  $t_1 \times t_2 \times \dots \times t_m$ .

The following proposition shows the relation of  $\text{sir}(\bigcup_{i=1}^m \overline{\text{ken}(I_i)})$  to  $I_1, I_2, \dots, I_m$  (we restrict our attention to the case  $m=2$ ).

Proposition 1: The parallel concatenation  $t$  of two traces  $t_1$  and  $t_2$  is a trace on  $\Sigma_1 \cup \Sigma_2$  with respect to

$$R = ((I_1 \cup I_2) \setminus (\Sigma_1 \cap \Sigma_2))^2 \cup (I_1 \cap I_2) \cup ((\Sigma_1 \setminus \Sigma_2) \times (\Sigma_2 \setminus \Sigma_1)) \cup ((\Sigma_2 \setminus \Sigma_1) \times (\Sigma_1 \setminus \Sigma_2)).$$

Proof:

It is because of  $\text{sir}(\overline{\text{ken}(I_1)} \cup \overline{\text{ken}(I_2)}) = R$ . Now, with  $\Sigma_1, \Sigma_2, \dots, \Sigma_m, I_1, I_2, \dots, I_m$  as above, let  $T_1, T_2, \dots, T_m$  be trace languages on  $\Sigma_1, \Sigma_2, \dots, \Sigma_m$  under  $I_1, I_2, \dots, I_m$  and  $L_1, L_2, \dots, L_m$  be languages on  $\Sigma_1, \dots, \Sigma_m$  respectively.

Definition 2: The parallel concatenation of  $T_1, T_2, \dots, T_m$  and the projective product of  $L_1, L_2, \dots, L_m$  (denoted by  $T_1 \times T_2 \times \dots \times T_m$  and  $L_1 \otimes L_2 \otimes \dots \otimes L_m$  respectively) are defined as follows:

$$T_1 \times T_2 \times \dots \times T_m = \{t \mid t = t_1 \times t_2 \times \dots \times t_m, t_i \in T_i, i = \overline{1, m}\},$$

$$L_1 \otimes L_2 \otimes \dots \otimes L_m = \cup \{w_1 \otimes w_2 \otimes \dots \otimes w_m \mid w_i \in L_i, i = \overline{1, m}\}.$$

It follows from theorem 1. that if  $T = [L]_I$  is trace language on  $\Sigma$  under  $I$  and  $\overline{\text{ken}(I)} = \{A_1, A_2, \dots, A_n\}$  then

$$\{T\} \subseteq \bigotimes_{i=1}^n h_i(L). \quad (*)$$

The trace language equating (\*) plays an important role in studying systems decomposable into sequential components. So we refer to "decomposable condition" as:

$$\{T\} = \bigotimes_{i=1}^n h_i(L).$$

This condition shall be concerned to in the next sections.

### 3. SYNTHESIZED COMPUTATION SYSTEMS AND THEIR BEHAVIOUR

In this section, computation systems take the general form presented in [3].

Definition 3: A computation system consists of:

- (i) a set  $D$  (states),
- (ii) an element  $x$  of  $D$  (the initial state),
- (iii) a finite set  $\Sigma$  of operations,
- (iv) a function "-" from  $\Sigma$  to the set of partial functions from  $D$  to  $D$ . The function - is extended to  $\Sigma^*$  in the usual way. We sometimes write  $S = (D, \Sigma, x)$  instead of  $S = (D, \Sigma, x, \bar{\quad})$ .

The set  $C_S$  of all computation sequences (from  $x$ ) and the reachability set  $R_S$  of reachable states of computation system  $S$  are defined as

$$C_S = \{\alpha \in \Sigma^* \mid \bar{\alpha}(x) \text{ is defined}\},$$

$$R_S = \{y \in D \mid \exists \alpha \in \Sigma^*, \bar{\alpha}(x) = y\}.$$

By a synthesized computation system, we think of computation one coming from these being concurrently active with some synchronization conditions. We shall confine our attention to the case when synchronization conditions come from the fact that some actions must take place at the same time. By constructing homomorphisms, we shall reduce that case to the one when the "contemporary" actions are common ones of some component systems. The following definition is consistent in that case:

Definition 4: Let  $S_i = (D_i, \Sigma_i, x_i, \bar{\phantom{x}}^{-i})$ ,  $i=1,2,\dots,n$  be computation systems. The synthesized computation system of  $S_1, S_2, \dots, S_n$  (denoted by  $S_1 \times S_2 \times \dots \times S_n$ ) is the following:

$$S = (D, \Sigma, x, \bar{\phantom{x}})$$

where

$$D = D_1 \times D_2 \times \dots \times D_n,$$

$$\Sigma = \Sigma_1 \cup \Sigma_2 \times \dots \times \Sigma_n,$$

$$x = (x_1, x_2, \dots, x_n), \text{ and}$$

$$\bar{\phantom{x}} : \Sigma \rightarrow (D \rightarrow D)$$

is defined as follows:

$$\forall \alpha \in \Sigma, \bar{\alpha}(y_1, y_2, \dots, y_n) = (z_1, z_2, \dots, z_n) \text{ iff:}$$

$$z_i = \bar{\alpha}^{-i}(y_i), \alpha \in \Sigma_i$$

$$z_i = y_i$$

in the other cases.

Now let  $h_i$ ,  $i=1,2,\dots,n$  be projection from  $\Sigma$  into  $\Sigma_i$ ,  $i=1,2,\dots,n$ . When  $\bar{\alpha}(x)=y$  we write  $x \xrightarrow{\alpha} y$  for convenience. The connection between the behaviour of  $S$  and the behaviour of  $S_1, S_2, \dots, S_n$  is showed by the following theorem:



Theorem 2:

$$C_S = C_{S_1} \otimes C_{S_2} \otimes \dots \otimes C_{S_n} .$$

Proof:

$$w \in C_{S_1} \otimes C_{S_2} \otimes \dots \otimes C_{S_n} \iff \forall i=1,2,\dots,n,$$

$$h_i(w) \in C_{S_i} \iff \forall i=1,2,\dots,n,$$

$\exists y_i \in D_i$  such that

$$h_i(w) \xrightarrow{i} y_i \iff w \in C_S$$

by the definition of "-" in S.

Remark: We sometimes deal with the set of computation sequences of a computation system, which lead the system to the state in the given set of states.

Denoting

$$C_S(>Q) = \{ \alpha \in \Sigma^* \mid \alpha \xrightarrow{\alpha} y, y \in Q \subseteq D \}$$

we have also: (by modifying consistently the proof of theorem 4):

$$C_S(>Q_1 \times Q_2 \times \dots \times Q_n) = C_{S_1}(>Q_1) \otimes \dots \otimes C_{S_n}(>Q_n),$$

where

$$Q_i \subseteq D_i, i=1,2,\dots,n.$$

Now we consider computation systems realized by Petri-nets [3]. We shall combine Petri-nets with one to another in the way presented in [5].

A Petri-net  $\underline{P} = (\Pi, \Sigma, \Delta, \alpha)$  consists of:

- (i) a finite set  $\Pi$  of places,
- (ii) a finite set  $\Sigma$  of transitions,
- (iii) an incidence function  $\Delta: \Pi \times \Sigma \cup \Sigma \times \Pi \rightarrow \{0,1\}$ ,
- (iv) an initial marking  $\alpha: \Pi \rightarrow N$ .

A function  $y: \Pi \rightarrow \mathbb{N}$  is called a marking. When  $\Pi = \{p_1, p_2, \dots, p_k\}$ , we sometimes regard a marking  $y$  as an  $n$ -dimensional vector  $\langle y(p_1), y(p_2), \dots, y(p_k) \rangle$ .

Let  $D_{\underline{P}}$  be a set of markings of  $P$ . For each  $a \in \Sigma$  a partial function  $\bar{a}: D_{\underline{P}} \rightarrow D_{\underline{P}}$  is defined as follows:

Let  $y \in D_{\underline{P}}$ . Then  $\bar{a}(y)$  is defined if and only if  $y(p_i) \geq \Delta(p_i, a)$  for all  $p_i \in \Pi$ . Suppose that  $\bar{a}(y)$  is defined, then

$$\bar{a}(y)(p_i) = y(p_i) - \Delta(p_i, a) + \Delta(a, p_i), \quad p_i \in \Pi.$$

The computation system  $S_{\underline{P}} = (\Sigma, D_{\underline{P}}, \bar{\cdot})$  is said to be realized by  $P$ .

The Petri-net  $\underline{P}$  is said to be safe if  $R_{S_{\underline{P}}} \subseteq \{0, 1\} \times \{0, 1\} \times \dots \times \{0, 1\}$ . When  $\underline{P}$  is a safe Petri-net, each marking  $y$  of  $R_{S_{\underline{P}}}$  can be written as a subset  $M$  of  $\Pi$ , (namely,  $y(p_i) = 1$  iff  $p_i \in M$ ).

Definition 4: Let  $\underline{P}$  be a safe Petri-net. A relation  $I$  on  $\Sigma$  is said to be an independency relation generated by  $\underline{P}$  iff:

$$\forall a, b \in \Sigma, (a, b) \in I \iff \exists \text{ marking } M \in R_{S_{\underline{P}}}$$

such that both  $a$  and  $b$  are enabled at  $M$  and  $\Delta(p, a) \cdot \Delta(p, b) = 0$ ,  $\forall p \in \Pi$  ( $a$  is called to be enabled at  $M$  if  $\forall p \in \Pi$ ,  $(\Delta(p, a) = 1) \Rightarrow y(p) \geq 1$ ).

Definition 5: Trace language  $T$  is said to be realized by safe Petri-net  $\underline{P}$  if  $T$  is a trace language on  $\Sigma$  under the independency relation generated by  $\underline{P}$  and  $\{T\} = C_{S_{\underline{P}}}$ .

Let  $\underline{P}_i = (\Pi_i, \Sigma_i, \Delta_i, X_i)$ ,  $i=1, 2$  be Petri-nets,  $S_1$  and  $S_2$  be the computation systems realized by  $\underline{P}_1$  and  $\underline{P}_2$  respectively. Assuming  $\Pi_1 = \{p_1, \dots, p_k\}$ ,  $\Pi_2 = \{p_{k+1}, \dots, p_m\}$ ,  $\Pi_1 \cap \Pi_2 = \emptyset$ . We consider the Petri-net  $\underline{P}_1 \times \underline{P}_2$  received from  $\underline{P}_1$  and  $\underline{P}_2$  in the following way [see 5]:

$$\underline{P} = \underline{P}_1 \times \underline{P}_2 = (\Pi, \Sigma, \Delta, x),$$

where

$$\Pi = \Pi_1 \cup \Pi_2, \quad \Sigma = \Sigma_1 \cup \Sigma_2, \quad \Delta(p_i, a) = \begin{cases} \Delta_1(p_i, a) & \text{if } a \in \Sigma_1, \\ 0 & \text{otherwise} \end{cases}$$

for  $i=1, 2, \dots, k$  and

$$\Delta(p_i, a) = \begin{cases} \Delta_2(p_i, a) & \text{if } a \in \Sigma_2, \\ 0 & \text{otherwise} \end{cases}$$

for  $i=k+1, k+2, \dots, m,$

$$x = (x_1, x_2).$$

Let  $S$  be the computation system realized by  $\underline{P}$ .

Theorem 3:

$$S = S_1 \times S_2.$$

Proof: Each marking of  $\underline{P}$  can be written as

$$(y_1, y_2), \quad y_1 \in N^k, \quad y_2 \in N^{m-k}.$$

By definition of  $\Delta$ ,  $\forall i=1, 2, \dots, k,$

$$y(p_i) \geq \Delta(p_i, a) \forall a \in \Sigma \iff y_1(p_i) \geq \Delta_1(p_i, a) \forall a \in \Sigma_1,$$

$\forall i=k+1, k+2, \dots, m,$

$$y(p_i) \geq \Delta(p_i, a) \forall a \in \Sigma \iff y_2(p_i) \geq \Delta_2(p_i, a) \forall a \in \Sigma_2.$$

That is,  $a$  is enabled in marking  $y$  of  $\underline{P}$  if and only if  $a$  is enabled in  $y_j$  when  $a \in \Sigma_j$ ,  $j=1, 2$ . Furthermore

$$\forall i = 1, 2, \dots, k$$

$$y(p_i) - \Delta(p_i, a) + \Delta(a, p_i) = \begin{cases} y_1(p_i) - \Delta_1(p_i, a) + \Delta_1(a, p_i) & \text{if } a \in \Sigma_1, \\ y_1(p_i) & \text{if } a \notin \Sigma_1. \end{cases}$$

$$\forall_i = k+1, k+2, \dots, m.$$

$$y(p_i) - \Delta(p_i, a) + \Delta(a, p_i) = \begin{cases} y_2(p_i) - \Delta_2(p_i, a) + \Delta_2(a, p_i) & \text{if } a \in \Sigma_2, \\ y_2(p_i) & \text{if } a \notin \Sigma_2. \end{cases}$$

Since that  $y \xrightarrow{a} y' \iff y_j \xrightarrow{a} y'_j$  when  $a \in \Sigma_j$  ( $j=1,2$ ) and  $y_j = y'_j$  when  $a \notin \Sigma_j$ , ( $j=1,2$ ). That means,  $S = S_1 \times S_2$

Corollary 2: If  $P_1$  and  $P_2$  are safe nets then  $P$  is a safe one.

Proof: Since  $R_S \subseteq R_{S_1} \times R_{S_2}$ .

Theorem 4: If  $T_1$  and  $T_2$  are trace languages realized by safe Petri-nets then so is  $T_1 \times T_2$ .

Proof: Let safe Petri-nets  $P_1, P_2$  be realisations of  $T_1, T_2$  respectively. By theorem 3 and corollary 2,  $P_1 \times P_2$  is safe Petri-net and  $C_{S_{P_1 \times P_2}} = \{T_1\} \times \{T_2\}$ . Of course,  $\{T_1\} \times \{T_2\} = \{T_1 \times T_2\}$ .

$$\forall t = t_1 \times t_2 \in T_1 \times T_2, \forall w \in t \implies [w]_I \subseteq t,$$

where  $I$  is the independency relation generated by  $P_1 \times P_2$ . This is followed from the fact that  $I \subseteq \text{sir}(\overline{\text{ken}(I_1)} \cup \overline{\text{ken}(I_2)})$ , where  $I_1$  and  $I_2$  are the independency relations generated by  $P_1$  and  $P_2$  respectively.

For every  $w' \in t$ ,  $w' \in [w]_{\text{sir}(\overline{\text{ken}(I_1)} \cup \overline{\text{ken}(I_2)})}$ , there exist

$$w_1, w_2, \dots, w_n, w_1 = w, w_n = w', w_i = w_i^1 a b w_i^2, w_{i+1} = w_i^1 b a w_i^2,$$

$(b, a) \in \text{sir}(\overline{\text{ken}(I_1)} \cup \overline{\text{ken}(I_2)})$ ,  $w_1 = w \in [w]_I$ . If  $w_i \in [w]_I$ , since

$w_i, w_{i+1} \in \{T_1\} \times \{T_2\}$ , there exists  $m \in R_{S_{P_1 \times P_2}}$  such that both  $a$  and  $b$  are enabled at  $m$ . On the other hand, from definition of  $P_1 \times P_2$  and proposition 1 it follows that if  $(a, b) \in \text{sir}(\overline{\text{ken}(I_1)} \cup \overline{\text{ken}(I_2)})$  then  $\forall p \in \Pi, \Delta(p, a) \cdot \Delta(p, b) = 0$ . This implies that, in our case,  $(a, b) \in I$  which means  $w_{i+1} \in [w]_I$ . The inductive principle gives  $w' \in [w]_I$  and this completes the proof of theorem 4.

This theorem states the closure property of the family of trace languages realized by safe Petri-nets under parallel concatenation.

Corollary 3: A trace language  $T = [L]_I$  satisfying the decomposable condition (in the 2 section) is realized by safe Petri-net. if for every  $i=1, 2, \dots, n$ ,  $h_i(L)$  is realized by safe Petri-net.

(The analogous and stronger result has been stated by E.Knuth in [2].)

Now we conclude this section by a small remark. Namely, synthesized computation systems defined in this paper, to our knowledge, are general enough to research the synchronization of asynchronised processes. In the definition of it, if  $\Sigma_1, \Sigma_2, \dots, \Sigma_{n-1}$  are disjoint pairwise and  $\Sigma_n = \Sigma_1 \cup \Sigma_2 \cup \dots \cup \Sigma_{n-1} = \Sigma$  then the systems  $S$  can be considered as the synchronization of asynchronised systems  $S_1, S_2, \dots, S_{n-1}$  by  $S_n$ . When  $S_n$  is realized by Petri-net,  $S$  turns to a multiprocessor system defined and studied detailly by P.H.Starke [7]. When  $S_n$  is a unshared producer-consumer system [3],  $S$  turns to a system synchronized by P-V operations. The same method used in studying those systems can be used to study our system also.

#### 4. ANALYSING SYNTHESIZED SYSTEMS VIA THEIR COMPONENTS

Many properties of synthesized systems can be received through the properties of their components. Unfortunately, those properties have been based on the regularity and it is not very useful to analyze synthesized systems by analyzing their components as the regularity is preserved by synthesizing.

As for us, we think that the most useful thing of this way is in verifying systems and proving the correctness of translating from one to another.

This section is devoted to the application of the above concept in the verification of synthesized systems. We shall take the method presented in [4].

Let

$$S = S_1 \times S_2 \times \dots \times S_n .$$

By [4], our task is construct an assertion system for S.

Assume that  $AS_1, AS_2, \dots, AS_n$  are assertion systems for  $S_1, S_2, \dots, S_n$  respectively,  $AS_i = (V_i, E_i, M_i)$ ,  $i=1, 2, \dots, n$ . We construct AS for S as follows:

$$AS = (V, E, M),$$

where

$$V = V_1 \times V_2 \times \dots \times V_n,$$

$$E = \{ ((v_1, v_2, \dots, v_n), t, (v'_1, v'_2, \dots, v'_n)) \mid \\ \text{if } t \in E_i, (v_i, t, v'_i) \in E_i, \text{ if } t \notin E_i, v_i = v'_i \},$$

$$M = M_1 \times M_2 \times \dots \times M_n: V_1 \times V_2 \times \dots \times V_n \rightarrow 2^D,$$

$$M(v_1, v_2, \dots, v_n) = M_1(v_1) \times M_2(v_2) \times \dots \times M_n(v_n) \subseteq \\ \subseteq D_1 \times D_2 \times \dots \times D_n .$$

Proposition 2: If  $AS_i$  are correct assertion systems for  $S_i$ , complete for  $V_i'$ ,  $i=1,2,\dots,n$  than  $AS$  is a correct asseccion system for  $S$  and complete for  $V'=V_1' \times V_2' \times \dots \times V_n'$ .

Proof: Denote

$$xt = \{y \mid \exists x \in X, x \xrightarrow{t} y\} \quad \text{for any } X \subseteq D.$$

We have

$$\forall ((v_1, v_2, \dots, v_n), t, (v_1', v_2', \dots, v_n')) \in E$$

$$M((v_1, v_2, \dots, v_n)) = M_1(v_1) \times M_2(v_2) \times \dots \times M_n(v_n) \neq \emptyset.$$

by  $M_i(v_i) \neq \emptyset \quad \forall i=1,2,\dots,n$ .

If  $t \notin \Sigma_i$  then  $v_i' = v_i$  and  $M_i(v_i) = M_i(v_i')$ . Since that

$$M((v_1, v_2, \dots, v_n)) \xrightarrow{t} Q = (Q_1', Q_2', \dots, Q_n')$$

where  $Q_i' = Q_i$  if  $t \in \Sigma_i$  and  $Q_i' = M_i(v_i)$  in the otherwise. It means that  $AS$  is correct for  $S$ .

To show that  $AS$  is complete for  $V'=V_1' \times V_2' \times \dots \times V_n'$ , we should note that  $\forall v' \in V', \forall x, y \in D, t \in \Sigma$ , if  $x \xrightarrow{t} y, x \in M(v')$  then  $x_i \xrightarrow{t} y_i$  in  $S_i$  when  $t \in \Sigma_i$  and  $x_i = y_i$  when  $t \notin \Sigma_i$ . Furthermore  $\forall i=1,2,\dots,n, x_i \in M_i(v_i')$ . Since  $\forall i=1,2,\dots,n, AS_i$  is complete for  $V_i'$ , there exist  $v_i \in V_i, (v_i', t, v_i) \in E_i$  for  $t \in \Sigma_i$ . Hence, putting  $v = \delta_1, \delta_2, \dots, \delta_n, \delta_i = v_i'$  if  $t \notin \Sigma_i$  and  $\delta_i = v_i$  if  $t \in \Sigma_i$  we have  $(v', t, v) \in E$ .

## 5. CONCLUSION

We have shown certain connections between trace languages and projective products. Mathematically, they are different from each to other, but both are introduced to for the purpose of studying the behaviour of concurrent computation systems, especially in representing their concurrency.

The approach presented in this paper can be used in studying concrete systems (such as distributed systems, multiprocessor systems) and the concurrency measure of synthesized systems.

## 6. REFERENCES

- |1| E.Knuth: Petri-nets and regular trace languages. April, 1978, The University of Newcastle upon Tyne, Computing Laboratory.
- |2| E.Knuth, Gy.Győry, L.Rónyai: A study of the projection operation. Application and theory of Petri-nets. Springer-Verlag, 1982.Vol.52.
- |3| Takumi Kasai and R.E.Miller: Homomorphisms between models of parallel computation. J. of Comp. and Syst.Sciences, 25, (1982).
- |4| Horst Müller: Inductive assertions for analysing reachability sets (in 2).
- |5| C.André: Behaviour of a place-transition net on a subset of transitions (in 2).
- |6| Ryszard Janicki: Nets, sequential components and concurrency relations. Theoret.Computer Science 29, (1984) 87-121.



- |7| Peter H. Starke: Multiprocessor systems and their concurrency. J. of Information Processing and Cybernetics - EIK - 20, (1984), 4.
  
- |8| A. Mazurkiewicz: Concurrent program schemes and their interpretations. DAIMI PB-78, Aarhus Univ. Press, 1977.

Megyjegyzések a nyomnyelvekről, projekciós és szintetizált  
számítási rendszerekről

DANG VAN HUNG

Összefoglaló

A szerző definiálja a párhuzamos szorzat fogalmát és rámutat bizonyos összefüggésekre a nyom-nyelvek és a projektív-szorzatok között. Megmutatja, hogyan lehet ezeket felhasználni a szintetizált számítási rendszerek elemzéséhez. A biztonságos Petri-hálókat és nyom-nyelvek egymáshoz való viszonyát is megvizsgálja.

Замечания о языках-следах, проекционных и синтетизированных  
вычислительных системах

Данг Ван Хунг

Резюме

Вводится понятие параллельного продукта и показывается связь между языками-следами и проективными продуктами. Показывается как могут использоваться эти понятия для анализа синтетизированных вычислительных систем. Рассматривается также связь между безопасными сетями Петри и языками-следами.

# A DATA STRUCTURE FOR QUADTREE CODES AND APPLICATION

PHAM NGOC KHOI

Institute of Technical Cybernetics  
SAV - Bratislava  
Czechoslovakia

## ABSTRACT

This paper describes a data structure for Quadtree code representation and the operators defined on it. Algorithms are presented for finding adjacent blocks, calculating geometric properties of region: area, perimeter and centroid. The efficiency of quadtree representation is evaluated in relation to Run length code and Chain code representation.

## 1. INTRODUCTION

For image data compression, a lot of representation algorithms is based on the various codings for a region: Run length codes, Chain codes, Quadtrees... [1,11]. Especially the tree representation which offers a number of advantages has attracted much attention in recent years [2-14]. A collection of algorithms has been studied for converting between quadtrees and other representations and measuring geometric features of regions represented by quadtrees [4-7, 13-14]. In this paper, we shall be concerned with a data structure for quadtrees as a tool for calculating region features of image. The emphasis is on algorithmic procedures, which are efficient especially from implementation point of view. The last section deals with the comparison of efficiency of image coding based on Quadtrees, Run length codes and Chain codes.

In the following by a binary image, we always mean a  $2^n \times 2^n$  array, each element of which has value 0 or 1, respectively to the color of pixel: black or white. The set of

1's pixels in a connected component of array is called a region.

## 2. DATA STRUCTURE FOR QUADTREES

Quadtree codes are an recent region representation method. It is based on successive subdivision of the array into quadrants until we obtain blocks possible simple pixel that are entirely contained in the region or entirely disjoint from it. Note that if the image is an  $2^n$  by  $2^n$  array of pixels, then after the  $k$ -th subdivision, each quadrant has  $2^{n-k}$  by  $2^{n-k}$  size. For example the region in Fig. 1a corresponds to raffinement process as shown in Fig. 1b. This process can be represented by a tree of degree 4 or a quadtree in which the entier array is a root node, the four sons of a node are its quadrants and the leaf nodes correspond to 1's or 0's pixel blocks and have their color BLACK or WHITE. The no-leaf nodes correspond to those blocks for which the further subdivision is still continued and have their color GRAY. The quadtree representation for Fig. 1b is shown in Fig. 1c. Note that here the blocks must have standard size and positions. Since the array was assumed to be size of  $2^n$  by  $2^n$ , the tree height is at most  $n$ . This region representation method was proposed by Klinger [2].

The quadtree can be defined as a tree whose nodes are either leaves or have four sons. A node can be gray, white or black and is, in general, stored as a record with six fields, five of which are pointers to the NW, NE, SW and SE quadrants and the sixth is a colors identifier. In the following, we propose a data structure advantageous to calculate some region features of image.

Assume that a square block has its four quadrants indexed as follows:

0	1
2	3

correspondently to quadrants NW, SE, SW, SE. Each quadrant in image is associated to a node in quadtree, which has four sons enumerated from left to right in the order 0,1,2,3. We present each node of quadtree a integer pair  $(L,K)$ , where:

- $L$  is level of node, i.e. distance from the root to the node,  $0 \leq L \leq n$
- $K$  is defined by

$$K = \sum_{i=0}^{L-1} n_i \cdot 4^i \quad 0 \leq n_i \leq 3$$

where  $(n_{L-1}, \dots, n_1, n_0)$  is the path from the root to the node  $P$ .

We denote

$$K = \overline{n_{L-1} \dots n_1 n_0}$$

$$P = (L, K)$$

$$Level(P) = L$$

$$Code(P) = K$$

We can use the following recurrent formula to determine the pair  $(L,K)$  for all nodes of a quadtree:

- The root has presentation  $(0,0)$
- If a node  $P$  is represented by  $(L,K)$  then its four sons have the representations

$$(L + 1, \quad 4K + i) \quad 0 \leq i \leq 3$$

This data structure was introduced by L.P. Jones and S. Iyengan [12] and is called virtual quadtree.

On this data structure we define the following operators:

1- "2's borrow" subtraction:  $Sub_2$

$$Sub_2 P = (L', K')$$

where

$$L' = L$$

$$K' = \text{sign } \overline{n'_{L-1} \dots n'_1 n'_0}$$

$n'_i$  are recursively computed as follows

$$- b_0 := 2$$

$$- n'_i := (n_{i+4} - b_i) \bmod 4$$

$$b_{i+1} := \begin{cases} 0 & \text{if } n_i \geq b_i \\ 2 & \text{otherwise} \end{cases} \quad i := 0, 1, \dots, L-1$$

$$\text{sign} := \begin{cases} + & \text{if } b_{L-1} = 0 \\ - & \text{if } b_{L-1} = 2 \end{cases}$$

For example:  $Sub_2(3, \overline{311}) = (3, \overline{133})$

$Sub_2(3, \overline{111}) = (3, \overline{-333})$

2- Clockwise rotation:  $Rot^+$

We first define operator  $Rot$  on the finite set  $\{0,1,2,3\}$

$$Rot^+(0) = 1$$

$$Rot^+(1) = 3$$

$$Rot^+(2) = 0$$

$$Rot^+(3) = 2 .$$

Now,  $Rot^+$  is naturally extended on the infinite set  $\{0,1,2,3\}^*$

$$Rot^+(xa) = Rot^+(x)Rot^+(a)$$

where  $x \in \{0,1,2,3\}^*$ ,  $a \in \{0,1,2,3\}$

Once again,  $Rot^+$  is extended on the set of quadtree nodes

$$Rot^+(L,K) = (L, Rot^+(K)) .$$

$Rot^+(P)$  gives the representation of the same node  $P$  after  $90^\circ$ -clockwise rotation of the image.

3- Counterclockwise rotation:  $Rot^-$

Operator  $Rot^-$  is defined in a similar manner to  $Rot$ , but it manipulates on the set  $\{0,1,2,3\}$  as follows

$$Rot^-(0) = 2$$

$$Rot^-(1) = 0$$

$$Rot^-(2) = 3$$

$$Rot^-(3) = 1 .$$

In the other word,  $Rot^- = (Rot^+)^{-1}$ .

$Rot^-(P)$  gives the representation of the same node  $P$  after  $90^\circ$ -counterclockwise rotation of image.

4- Rounding: *Rnd*

$$Rnd(P) = (L', K')$$

where

$$L' = L - 1$$

$$K' = \overline{n_{L-1} \dots n_2 n_1}$$

*App(P, i)* determines *i*-th son of *P* in Quadtree.

### 3, CALCULATING SOME PROPERTIES OF REGION

#### 3.1. ADJANCENCY

In connection with the adjacency of blocks, we have the theorem [13]:

Theorem: Given a variable *side* in the set {Northern, Western, Eastern, Southern}, which is encoded respectively by {0,1,2,3}. Let *P* be a quadrant of image. The quadrant *Q* is determined by

$$Q = Rot^{-side}(Sub2(Rot^{+side}(P)))$$

Thus if  $sign(Code(Sub2(Rot^{+side}(P)))) > 0$  then *Q* is the quadrant which is adjacent to *P* in the *side* direction and which has same size as *P*.

Basing on the theorem, we can have the procedure finding all quadtree leaves adjacent to a given leaf *P* in a given direction *side*



```
procedure JOINBLOCK (P,side,JOIN)
  /* input P: leaf of quadtree
     side: direction
     output JOIN : set of all leaves adjacent to P in
     direction side */
  node P,Q
  integer side
  set of node JOIN
  quadtree QTREE
  begin JOIN :=  $\emptyset$  ;
        Q := Rot-side(Sub2(Rot+side(P)));
        SEARCH(Q,QTREE,side,JOIN)
  end / joinblock /

procedure SEARCH(Q,QTREE,side,JOIN)
  node Q
  integer side
  set of node JOIN
  quadtree QTREE
  begin if Code(Q) < 0 then EXIT /* not adjacent block */
        else if Q not in QTREE then
          begin SEARCH(Rnd(Q),QTREE,side,JOIN1);
                JOIN := JOIN+JOIN1
          end
        else if Color(Q)  $\neq$  GRAY then JOIN := Q
        else
```

```
begin
SEARCH(App(Q, Rot+side(2)), QTREE, side, JOIN1);
JOIN := JOIN + JOIN1;
SEARCH(App(Q, Rot+side(3)), QTREE, side, JOIN1);
JOIN := JOIN + JOIN1
end

end /* search */
```

This algorithm can be efficiently used in some problems such as tracing the boundary of region, calculating the perimeter of region ...

### 3.2. AREA AND PERIMETER OF REGION

In order to calculate the perimeter of region, we visit, say, in postorder all black leaves of the quadtree. For each of them, we find all white leaves adjacent to it and the boundary segments. The sum of boundary segments yields the perimeter of region. The area of region is simply sum of area of all black leaves.

```
procedure GEOM(QTREE, n, PERI, AREA)
/* image is of size  $2^n \times 2^n$  */
node P, Q
quadtree QTREE
integer PERI, AREA
begin
P := (0,0)

/* find a black or white block in upper-left
corner */
while Color(P) = GRAY do P := App(P,0);
PERI := 0; AREA := 0
repeat /* cycle calculating perimeter and area */
while P not in QTREE do P := Rnd(P)
while Color(P) = GRAY do P := App(P,0)
```

```

    if Color(P) = BLACK do
begin
    AREA := AREA+(n-Level(P))2;
    for i := 0 to 3 do
begin
    JOINBLOCK (P,i,JOIN) ;
    if JOIN = ∅ then PERI := PERI+(n-Level(P))
    else for all Q in JOIN do
        if Color(Q) = WHITE then
            PERI := PERI+min(n-Level(P),
                n-Level(Q))
        end
    end
    P := (Level(P),Code(P)+1);
until Code(P) = 4n-Level(P)
end /* geom */

```

It is easy to see that the average execution time of the algorithm is proportional to the number of leaf nodes in quadtree.

### 3.3. CENTROID

The centroid of a binary image is a point  $(\bar{x}, \bar{y})$  such that  $\bar{x}$  is the average value of the  $x$ -coordinates of all the black points of the image and  $\bar{y}$  is the average of the  $y$ -coordinates of the black points. In other words, if there are  $m$  black points in the image  $(x_1, y_1), \dots, (x_m, y_m)$ , the centroid is

$$(\bar{x}, \bar{y}) = (\sum x_i / m, \sum y_i / m)$$

```

procedure CENTROID (QTREE, n, XCENT, YCENT)

```

```

/* calculate the centroid of quadtree for image of size
2n × 2n, XCENT, YCENT are the centroid value */

```

```
node P
integer n
quadtree QTREE
begin
  P := (0,0);
  XCORD := 0; YCORD := 0;
  MOMENT (P,n,XCORD,YCORD,X,Y,MASS);
  if MASS = 0 then begin
    XCENT := 0;
    YCENT := 0
  end
  else begin
    XCENT := X/MASS;
    YCENT := Y/MASS
  end
  and /* centroid */
```

```
procedure MOMENT (P,n,XCORD,YCORD,X,Y,MASS)
/* calculate the moments of order 0 and 1 for block P
XCORD, YCORD are are coordinates of upper-left corner of P
X is the moment of order 1  $m_{10}$ 
Y is the moment of order 1  $m_{01}$ 
MASS is the moment of order 0  $m_{00}$  */
node P
integer n,XCORD,YCORD,X,Y,MASS
begin
  X := 0; Y := 0; MASS := 0;
  if Color P = GRAY then for i := 0 to 3 do
    begin
      MOMENT (App(P,i),n,XCORD+2↑(n-Level(P)-1)*i mod 2,
              YCORD+2↑(n-Level(P)-1)*i div 2,X1,Y1,M1);
      X := X+X1; Y := Y+Y1; MASS := MASS+M1
    end
  else if Color(P) = BLACK then
    begin
```

```

X := (2*XCORD+2↑(n-Level(P))-1)*2↑(2*(n-Level(P))-1);
Y := (2*YCORD+2↑(n-Level(P))-1)*2↑(2*(n-Level(P))-1);
MASS := 2↑(2*(n-Level(P)))
end
end /* moment */

```

It is to see, once again, that in the procedure each black leaf in the tree is visited once and only one. The other moments can be calculated in an analogous way.

#### 4. EFFICIENCY OF QUADTREE REPRESENTATION

This section is devoted to the discussion of the efficiency of quadtree representation in relation to the other codings. Basing on quadtree codes we give the evaluations of storage space for run length codes, chain codes. Assume that an image of size  $2^n \times 2^n$  is represented by quadtree  $Q$  with  $Q$  nodes. Each node  $P$  of has 3 attributes:  $Level(P)$ ,  $Code(P)$  and  $Color(P)$ , which need respectively  $\log_2 n$  bits,  $2n$  bits and 2 bits to store them in memory space. Thus, the amount of storage needed for  $Q$  is

$$Q(2+2n+\log_2 n)$$

If  $2n$  bits for  $Code(P)$  have the representation

$$Code(P) = y_n x_n \dots y_1 x_1, \quad x_i, y_i = 0, 1$$

then the coordinates  $(x, y)$  of upper-left corner of  $P$  is determined as follows [8, 10]

$$x(P) = \sum_{i=1}^L x_i 2^{n-Level(P)+i-1}$$

$$y(P) = \sum_{i=1}^L y_i 2^{n-Level(P)+i-1}$$

and the size of  $P$  is

$$s(P) = 2^{n - \text{Level}(P)}$$

#### 4.1. EFFICIENCY OF QUADTREE REPRESENTATION IN RELATION TO RUN LENGTH CODE

Let  $B \subset Q$  be the set of black levels of quadtree. We have the integer set  $BY$  defined as follows:

$$BY = \{y(P) \mid P \in B\} \cup \{y(P) + s(P) + 1 \mid P \in B\}.$$

With the equivalence relation "=" in the usual sense on  $BY$ , we have the set  $\overline{BY} = BY / \equiv$ . Denote by  $b$  the cardinality of  $\overline{BY}$ . Each  $\bar{y}_j \in \overline{BY}$  determines the image row where may be arise a new run configuration (Fig. 2.)

In order to determine the number of runs, for each  $\bar{y}_j \in \overline{BY}$ , the associated set  $H_j \subset B$  and the relation  $\gamma_j \subset H_j \times H_j$  are defined as follows

$$H_j = \{P \in B \mid \bar{y}_j \in [y(P), y(P) + s(P)]\}$$

$$P_m \gamma_j P_1 \iff x(P_m) = x(P_1) + s(P_1) + 1$$

$$\text{or } x(P_1) = x(P_m) + s(P_m) + 1$$

$$P_m, P_1 \in H_j$$

Next, if let  $\gamma^*$  be the reflexive, transitive closure of  $\gamma_j$  on  $H_j$ , then we have the set  $H_j^* = H_j / \gamma^*$ , each element of which contains the blocks successively adjacent in horizontal direction.

The total number of runs in run length representation will be

$$\sum_{j=1}^b (\bar{y}_{j+1} - \bar{y}_j) \text{card } H_j^* \quad \text{where } \bar{y}_{b+1} = 2^n$$

Thus, the amount of run length code storage needed for image is

$$2(n+1) \sum_{j=1}^b (\bar{y}_{j+1} - \bar{y}_j) \text{card } H_j^* \text{ bits}$$

Now we can conclude that given a region represented by quadtree with  $Q$  nodes, run length code will be used efficiently when

$$\sum_{j=1}^b (\bar{y}_{j+1} - \bar{y}_j) \text{card } H_j^* < \frac{Q(2+2n+\log_2 n)}{2(n+1)}$$

#### 4.2. EFFICIENCY OF QUADTREE CODE REPRESENTATION IN RELATION TO CHAIN CODE

For each  $B$  in the set of black leaves  $\mathcal{B}$ , by the procedure JOINBLOCK we can determine the set

$$N_B = \{P \in \mathcal{B} \mid P \text{ is adjacent to } B\}$$

The boundary part, which belongs to  $B$  is calculated by

$$4(n-\text{Level}(B)) - \sum_{P \in N_B} \min(n-\text{Level}(B), n-\text{Level}(P))$$

Therefore the number of directional vectors in chain representation is

$$4 \sum_{B \in \mathcal{B}} (n-\text{Level}(B)) - \sum_{B \in \mathcal{B}} \sum_{P \in N_B} \min(n-\text{Level}(B), n-\text{Level}(P))$$

We obtain the formula calculating the amount of chain code storage needed

$$3(4 \sum_{B \in \mathcal{B}} (n-\text{Level}(B)) - \sum_{B \in \mathcal{B}} \sum_{P \in N_B} \min(n-\text{Level}(B), n-\text{Level}(P)))$$

At this point, we can arrive to the conclusion that given a region represented by quadtree, chain code will be used efficiently when

$$4 \sum_{B \in \mathcal{B}} n\text{-Level}(B) = \sum_{B \in \mathcal{B}} \sum_{P \in N_B} \min(n\text{-Level}(B), n\text{-Level}(P)) < \\ < \frac{Q(2+2n+\log_2 n)}{3}$$

## 5. CONCLUSION

For describing quadtree representation of region, a data structure has been presented with the operators defined on it. This operators can be easily implemented in usual programming languages. Based on the data structure, the algorithms have been described for finding adjacent blocks, calculating geometric features of region: area, perimeter and centroid. These algorithms are useful still in the case where a region may have holes. Basing on the data structure, we can study the algorithms of converting quadtree into chain codes and run length codes [13,14]. The last section deals with the analysis of representation efficiency of run length codes and chain codes in relation to a given quadtree. The explicite formulas have been presented for evaluating storages space memory needed for representations. By simple procedures, the calculation of these formulas can be done efficiently.

## REFERENCES

- [1] E.L. Hall: Computer image processing and recognition. New York - Academic Press 1979.



- [2] N. Alexandiris and Klinger: A picture decomposition, tree data structure and identifying directional symmetries as node combinations. CGIP Vol8 1978, p. 43-77
- [3] A. Rosenfeld: Quadrees and Pyramids for Pattern Recognition and Image Processing. IEEE 1980 5-th Int. Conference on PR. Miami Beach 1980, p. 802-807.
- [4] H. Samet: Region representation: Quadrees from boundary codes. C ACM Mar 1980 p. 163-170
- [5] C.R. Dyer et al.: Region representation: Boundary codes from Quadrees. C ACM Mar 1980 p. 171-179
- [6] M. Shneier: Calculation of Geometric Properties Using Quadrees. CGIP Vol16 N<sup>o</sup>3 July 1981, p. 296-302
- [7] H. Samet: Computing Perimeters of Regions in Image Represented by Quadrees. IEEE Trans. on PAMI, Vol PAMI-3, N<sup>o</sup>6, Nov 1981 p. 683-687
- [8] I. GARGANTINI: An Effective Way to Represent Quadrees. C ACM Vol 25 N<sup>o</sup>12 Dec 1982, p. 905-910
- [9] H. Samet: Neighbour finding Techniques for Images Represented by Quadrees CGIP 1982, p. 37-57
- [10] S.X. Li and M.H. Loew: Quadcodes and their application in Image Processing. George Washington University, Department of EE&CS Rept. N<sup>o</sup>IIST 83-15. Oct. 1983
- [11] G. Ram: On the Encoding and Representing of Images. CGIP 26 1984, p. 224-232
- [12] L.P. Jones and S.S. Iyengar: Space and Time Efficient Virtual Quadrees IEEE Trans. on PAMI Vol PAMI-6, N<sup>o</sup>2 Mar 1984, p. 244-248

- [13] P.N. Khoi and H. Kiem: Code conversion in image processing. Institute of Technical Cybernetics. SAV-Bratislava. Rept. N<sup>o</sup>14/1984
- [14] P.N. Khoi et al: Conversion and Synthesis of Image Representation. Proceeding of the Conference on Applied Mathematics. Hanoi 1985 (to appear)

A "négyzetes fa" /quadtrees/ kódok adatstruktúrája alkalmazásokkal

PHAM NGOC KHOI

Összefoglaló

A szerző a "négyzetes fa"-kódok reprezentációja számára bevezet egy adatstruktúrát és operátorokat definiál rajta. Algoritmusokat ad meg a kiegészítő blokkok megkeresésére, valamint a tartományok geometriai jellemzői /pl. terület, kerület, súlypont/ meghatározására.

Összehasonlítva a "futás-hosszuság" és "lánc" kód reprezentációkkal, értékeli a "négyzetes fa" reprezentáció hatékonyságát.

Структура данных для "квадратического дерева"-кодов с применением

Пхам Нгоц Кхой

Резюме

В статье описывается структура данных для представления кодов типа "квадратических деревьев" и определены операторы на этой структуре. Даются алгоритмы для нахождения дополнительных блоков и для вычисления геометрических свойств /площадь, периметр, центроид/ областей.

Представление типа "квадратических деревьев" сравнивается с представлением типа "длина пробега" и "цепь".



## AN EFFICIENT SYNTHESIS OF IMAGE MATCHING ALGORITHMS

HOANG KIEM, PHAM NGOC KHOI

Institute of Informatics and Cybernetics  
Hanoi-Vietnam

## ABSTRACT

In order to improve the efficiency of image matching, a lot of matching schemes have been proposed, based on various approaches [1-5]. Here we discuss the efficiency of the synthesis of image matching algorithms using hierarchical schemes and those that use the combination of coarse-fine matching algorithms. The method of extracting the features for regions in an image and performance of scene matching methods are considered.

This paper consists of the following parts:

- On the synthesis of image matching algorithms
- k-centroid feature extraction for image matching
- Combination of image transformation and normalization
- Synthesis scheme for image matching programs

## 1. ON THE SYNTHESIS OF IMAGE MATCHING ALGORITHMS

It is well known that the problem of scene matching is given a template of a scene, determine the location of this template in another scene. The method used to solve this problem, in its simplest form, is called template matching with the basic correlator, the statistical correlator. Later, some modifications using invariant moments for scene matching have been developed to solve the general problem involving geometrical and sensor variation [4-5].

Since a template of size  $M \times M$  can be shifted into  $(N-M+1)^2$  possible positions in an image of size  $N \times N$ , the number of correlations can be extremely large. The tendency in the current research is toward the use of hierarchical techniques for decreasing the number of search positions. In particular, coarse-fine techniques are logarithmically efficient and reduce the number of search positions to  $K \cdot \log(N-M+1)^2$ , where  $K$  is a constant. [1,2,3].

However, at level of search, the number of computations needed to obtain the features for scene matching for example, invariant moment can be still large. Later, a synthesis using hierarchical technique and detections is proposed.

### 1.1 Hierarchical schemes for image matching

- At first, a structured set of pictures at different resolution is defined. The level  $K$  scene is reduced to a level  $(K-1)$  scene with the agglomerative rule, for example:

$$F_{K-1}(i, j) = \frac{1}{4} \{ F_K(2i, 2j) + F_K(2i, 2j+1) + F_K(2i+1, 2j) + F_K(2i+1, 2j+1) \}$$

where,  $F_K(i, j)$  - the gray scal of pixel  $(i, j)$  at level  $K$ . Note that, at the level  $K$ , number of possible test locations is  $[(N-M+1)/(2^K+1)]^2$  and at level  $K-1$ , only the locations selected in level  $K$  needed to be tested.

- A matching rule to guide the search from level  $K-1$  to level  $K$  must also be defined. In the scene matching with invariant moments, this rule is the moment correlation which is costly in computation, due to the calculations needed to obtain the invariant features. But it can be used to great advantage at the low resolution level at which other methods are not possible. Here, we use an approach as follows: instead of matching each template of scene at every location, the templates are partitioned into "informative" and "irrelevant" templates by some simple tests. Elimination of mismatching locations and termination

of computation can take place at each level of test based on this partition.

In practice, we have used a detection that combined two simple tests before matching the scene with the invariant moments:

1<sup>o</sup>. Test based on measure of the similarity of two gray level distributions ( $\tau$ -test).

2<sup>o</sup>. Test based on the correlation coefficient of the joint distributions ( $\rho$ -test). The  $\tau$ , and  $\rho$  measures are computed for each location. If both  $\tau, \rho$  are smaller than selected threshold, this location is rejected.

- Thus, let  $N_k^i$  be a set of test locations  $(u, v)$  at search level  $K$ , with a matching rule  $R_k^i$  such that

$$N_k^i = \{(u, v) | R_k^i(u, v) \geq \theta_k^i, 1 \leq u, v \leq M\}$$

where  $\theta_k^i$  is the threshold selected to be used at search level  $K$ ,  $R_k^i$  is some matching rule at test location  $(u, v)$ ,  $M$  is the number of picture elements in the template. We can divide  $R_k^i$  into the preliminary rule detection by simple test and the main rule (for example, the moment correlation rule).

Let  $N_k := \bigcap_i N_k^i$ , for a search region of size  $N \times N$ , an  $(2N-2M+1)^2$ -matrix  $G_{k-1}$  was generated by

$$G_{k-1}(2i, 2j) = \begin{cases} 1 & \text{if } (i, j) \in N_k \\ 0 & \text{if } (i, j) \notin N_k \end{cases}$$

All other entries of  $G_{k-1}$  are set to zero. Tests are to be performed at the test locations for  $G_{k-1}(u, v) = 1$ . The search continues until one of two conditions is encountered

1<sup>o</sup>. At search level  $L=n$ ,  $G_n(u, v) = 1$  for one value of  $(u, v)$ , location  $(u, v)$  is declared the matched location.

2<sup>o</sup>. At the level  $L=0$ , there exist several locations  $(u, v)$  such that the  $G_0(u, v) = 1$ . Select the location with the highest correlation with the invariant moments.

1.2 Theorem

The condition for saving the computation time using this synthesis is following:

$$\Phi' < \Phi(1-p)$$

where  $\Phi'$  - the computation complexity of the detection at each location defined by the number of calculation needed.

$\Phi$  - the computation complexity of the main-matching rule at each location defined by the number of calculation needed.

$p$  - the probability of matching by the detection.

Proof:

Noting that, at level  $k$ , number of possible test locations is  $\lceil (N-M+1)/(2^k+1) \rceil^2$  then the number of calculation needed for scene matching using this synthesis is

$$\mathcal{N}_s = \lceil (N-M+1)/(2^k+1) \rceil^2 \cdot \Phi' + \lceil (N-M+1)/(2^k+1) \rceil^2 \cdot \Phi \cdot p$$

For saving the computation time using this synthesis, the following condition need to be satisfied:

$$\mathcal{N}_s < \lceil (N-M+1)/(2^k+1) \rceil^2 \Phi$$

It follows that

$$\begin{aligned} \lceil (N-M+1)/(2^k+1) \rceil^2 \cdot \Phi' + \lceil (N-M+1)/(2^k+1) \rceil^2 \Phi \cdot p \\ < \lceil (N-M+1)/(2^k+1) \rceil^2 \Phi \end{aligned}$$

Dividing both sides of the above inequality to  $\lceil (N-M+1)/(2^k+1) \rceil^2$  we have

$$\Phi' + \Phi \cdot p < \Phi$$

or  $\Phi' < \Phi(1-p)$



which proves the theorem.

Theoretical analysis and simulation with  $\tau$ -test and  $\rho$ -test in scene matching by invariant moments indicated that a saving of computation time as well as a high degree of precision in locating a region is possible.

### 1.3 The $\tau$ -test

The  $\chi^2$ -test measures the difference between two frequency distributions. Let  $h_m(k)$  be the frequency distribution of gray-scale intensities in a model window. Let  $h_t(k)$  be the frequency distribution of a test window. The significance of the  $\chi^2$ -test depends on the number of samples:

$$\chi^2 = \sum_k \frac{(h_m(k) - h_t(k))^2}{h_t(k)}$$

where, we can consider  $h_m$  to be a hypothetical ideal distribution. Let  $\tau = e^{-\chi^2/c}$ , where  $c$  is some positive constant.  $\tau$  is a measure of the similarity of two distributions (in the  $\chi^2$  sense) if the distributions are identical, then  $\tau$  will be unity, if they are very different,  $\tau$  will be close to zero.

$\tau$  is not sensitive to the location of pixels. It simply measures the degree of similarity between two marginal distributions.

### 1.4 The $\rho$ -test

Let  $m_1$  be the mean of  $h_m$  and  $m_2$  be the mean of  $h_t$ . Let  $\sigma_1$  be the standard deviation of  $h_m$  and  $\sigma_2$  be the standard deviation of  $h_t$ . We define the coefficient  $\rho$  as follows:

$$\rho = \frac{\mu(1,1)}{\rho_1 \cdot \rho_2}$$

where,  $\mu(i,j) = \frac{1}{n} \sum_k (x_m(k) - m_1)^i \cdot (x_t(k) - m_2)^j$ .

$x_m(k)$  and  $x_t(k)$  are

are gray values of the  $k$ -th pixel in the model image and the test image, respectively.

$\rho$  is in the interval  $[-1, 1]$ . In general, if there is a linear functional dependence between the test and model window  $\rho$  will be 1. If the window are independent distributions  $\rho$  will be 0. Thus, the intermediate values will measure the degree of dependence between the two windows.

$\rho$  is a good test for the proper location of pixels. With the systematic change in lighting,  $\tau$  would be small but  $\rho$  would be large because the test and model distributions would still be well-correlated.

### 1.5 Performance of scene matching methods

To evaluate the performance of any of matching techniques one may consider the probability distribution that could be at the  $k$ -th level in the hierarchical search or the first level for template matching. The distribution  $p_k(R)$  is the probability that the true match location takes on a specific similarity value  $R$ .  $R_k(u^*, v^*)$  is the similarity value at the true math location for a particular match under consideration. Let  $P_k$  be the probability of detection of the  $k$ -th search level (i.e. the probability that the similarity measure at the true match location exceed the threshold  $R_T^k$ ) and the  $p_k(R)$  will be assumed to have a Gaussian distribution with a variance of  $(\sigma_R^k)^2$ . Then  $P_k$  can be expressed as

$$P_k = \int_{R_T^k}^{\infty} p_k(R) dR = \frac{1}{\sqrt{2\pi} y} \int_{R_T^k}^{\infty} \exp\left(-\frac{R^2}{2y^2}\right) dR$$

where

$$y = [R_T^k - R_k(u^*, v^*)] / \sigma_R^k$$

Similarly, the probability of false fix at the  $k$ -th search level can be computed by

$$P_f = \int_{R_T^k}^{\infty} p_f(R) dR = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^y \exp\left(-\frac{R^2}{2}\right) dR = 1 - \Phi(y)$$

where  $y = (R_T^k - R_b^k) / \sigma_f^k$ ,  $p_f(R)$  is the probability density distribution of the similarity measures of all test locations except the true match location with a variance of  $(\sigma_f^k)^2$ ,  $R_b^k$  is the similarity measure averaged over all test locations (Fig.1)

The error introduced by the Gaussian assumption for a non-Gaussian distribution may be large in the case of small values of  $p_k(R)$ . We can use the Edgeworth expansion for this case.

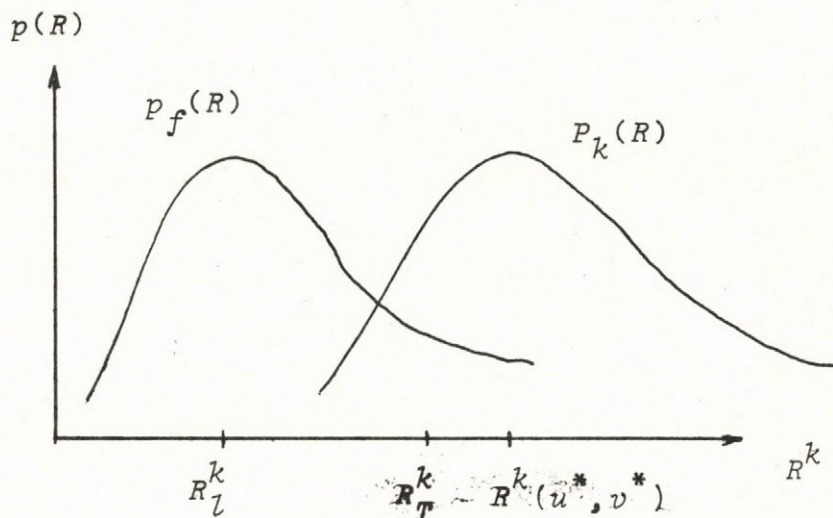


Figure 1. Probability density function at match location .  $P_k(R)$  and back-ground  $p_f(R)$

$\sigma_R^k$  obtained on that level to achieve a given probability of a match.

To select the threshold  $R_T^k$ , the invariant moments of the image were correlated with the moments of the image reduced by a factor of 2 and rotated by  $2^\circ$  and  $45^\circ$ . The averaged correlation of the three cases was then used as a bound to estimate a threshold sequence. Noting that the correlation were made on the logarithms of amplitude of the moments rather than the amplitude itself.

## 2. FEATURE EXTRACTION FOR IMAGE MATCHING

At the fine-matching step, the moment approach is often used for this purpose. However, scene match, with invariant moments is costly in computation. In many cases, the following approaches give us the powerful features for image matching with smaller computation time.

### 2.1 The kG-centroid (kL-centroid) features

We define a kG-centroid (kL-centroid) of the image as a centroid of this image at gray level  $k$  (of  $k$ -th region of this image). Suppose that  $(x_c^k, y_c^k)$ ,  $k=1, 2, \dots, K$  are kG-centroid (kL-centroid) and  $(x_c, y_c)$  is centroid of the image  $I$ .

Let  $(r^k, \theta^k)$  be polar coordinates of  $(x_c^k, y_c^k)$  in the polar coordinate system with  $(x_c, y_c)$  as the origin and the rayon passed  $(x_c, y_c)$ ,  $(x_c^k, y_c^k)$  as the initial rayon  $k^0$  is a chosen value, say,  $k^0 = 1$ .

By converting  $(r^k, \theta^k)$  into  $(\tilde{r}^k, \theta^k)$ , where

$$\tilde{r}^k = r^k / \sum_{i=1}^k r^i$$

then  $(\tilde{r}^k, \theta^k)$ ,  $k=1, 2, \dots, K$ , are invariant features in relation to translation, rotation and size change.

We can also derive invariant features from kG-centroid (kL-centroid) in the following manner.

Let  $\rho(k, k')$  be the distance between  $(x_c^k, y_c^k)$  and  $(x_c^{k'}, y_c^{k'})$  where  $k, k'=1, 2, \dots, K$ , then  $\{\rho(k, k')\}_{k, k'=1, 2, \dots, K}$  are dependent only on the shape of the object, but not affected by its location, orientation or relative size.

For the normalized images, we can extract simpler features as follows.

## 2.2 Projections and Cross-Sections

Given a two-dimensional continuous function  $f(x,y)$  we define the projection of  $f(x,y)$  in the  $x$  and  $y$  direction are

$$\int_R f(x,y)dx \quad \text{and} \quad \int_R f(x,y)dy$$

In principle, projections in a sufficient number of directions contain enough information to reconstruct the picture [5].

For a digital image, the  $x(i)$  and  $y(j)$  projections are defined as:

$$x(i) = \sum_j f(i,j)$$

$$y(j) = \sum_i f(i,j) \quad \text{for} \quad 1 \leq i, j \leq N$$

More detailed information about the arrangement of gray levels in the region  $R$  can be obtained by using projections of  $f(x,y)$  in various directions.

In many cases, the projections on X-Y axes a certain amount of information of object for recognition and the number of dimension may be significantly reduced from  $N^2$  to  $2N$ , ( $N$  is the dimension of the image). Furthermore, the numerical properties of projections, such as their (one-dimensional) moments, Fourier coefficients, Walsh coefficients, etc... can be used as the powerful features for recognition of the image in which, most of the strokes of patterns are either horizontal or vertical and they generate many step segments in the projections. Some experiments have been made successfully by these features [13].

## 3. COMBINATION OF IMAGE TRANSFORMATION AND NORMALIZATION

As we know that, the power spectrum of an image is to be independent of translation. The Mellin transform has been show to be scale independent. The Polcar-Cartesian transform converts rotation into translation. Hence a combination of these performed

successively will allow shape to be matched to shape independent of translation, rotation and scale [10,15].

Here we use a simple normalization scheme, the normalized image which also is invariant to object translation, rotation and size change.

- Normalization in relation to translation

The image is normalized to an image-centered coordinate system with its centroid is translated into a fixed point

$$N(x,y) = I(x_0+x-x_c, y_0+y-y_c)$$

where,  $(x_0, y_0)$  be a fixed point,  $(x_c, y_c)$  be the centroid of image.

- Normalization in relation to rotation

The image is normalized to coordinate system with its principal axes as coordinate axes

$$N(x,y) = I(x \cdot \cos\theta - y \cdot \sin\theta, x \cdot \sin\theta + y \cdot \cos\theta)$$

where

$$\theta = \frac{1}{2} \tan^{-1} \frac{2 \cdot I_{11}}{I_{20} - I_{02}}, \quad I_{pq} = \sum_s (x-x_c)^p (y-y_c)^q f(x,y)$$

- Normalization in relation to size change

The image is scaled to a standard size

$$N(x,y) = I(k_x \cdot x, k_y \cdot y)$$

where  $(k_x, k_y)$  be the ratio of the size of the image  $I$  to standard size.

In this way, the normalized image dependent only on the shape of the object, but not affected by its location, orientation, or relative size.

#### 4. SYNTHETIC SCHEME FOR IMAGE MATCHING PROGRAMS

In connection with scheme of synthesis of image matching algorithms (Fig. 3) we propose an use manner of programs as follows.

Given an image of size  $N \times N$  and a template of size  $M \times M$ , if necessary, we make a geometrical correction for input image by using GECOR (See appendix). The next phase is coarse matching the template to windows of image. The proposed step matching is  $M/2$ . In this phase, we combine several tests ( $\tau$ -test,  $\rho$ -test) by using HISTO, PROFIL, THRSLD (See appendix). So we obtain the possible match locations.

In the next phase, with each of the possible match locations, we make a fine matching around those locations. For improving the efficiency of fine matching, both the template and the window may be modified either by transforms Polcar, Mellin, Fourier using TPOCAR, MELIN,  $TF\phi$  (See appendix) or by normalizations: centered translation, rotation, scaling using TCEN, TROTA,  $SCALE\phi$  (See appendix). With these transformations (normalizations), the transformed (normalized) template and window will be invariant to rotation, translation, scale... After that, we can use the simple features such as projections for matching. Then the fine matching may be made by computing the feature correlation: invariant moments, kL-centroid (kG-centroid), profils...using MOMENT, TOPO, PROFIL, THRSLD (See appendix). We will present program descriptions in the appendix.

#### 5. CONCLUSIONS

Experimental results indicated that scene matching with the basic sequential method provided good performance in the matching of scene that contain relatively well-defined objects of varying background. This method is particularly useful in matching images taken by the same type of sensors under different operating conditions. Depending on the scene content, scene matching with invariant moments was successful in some cases. In particular, this method can be used to great advantage at

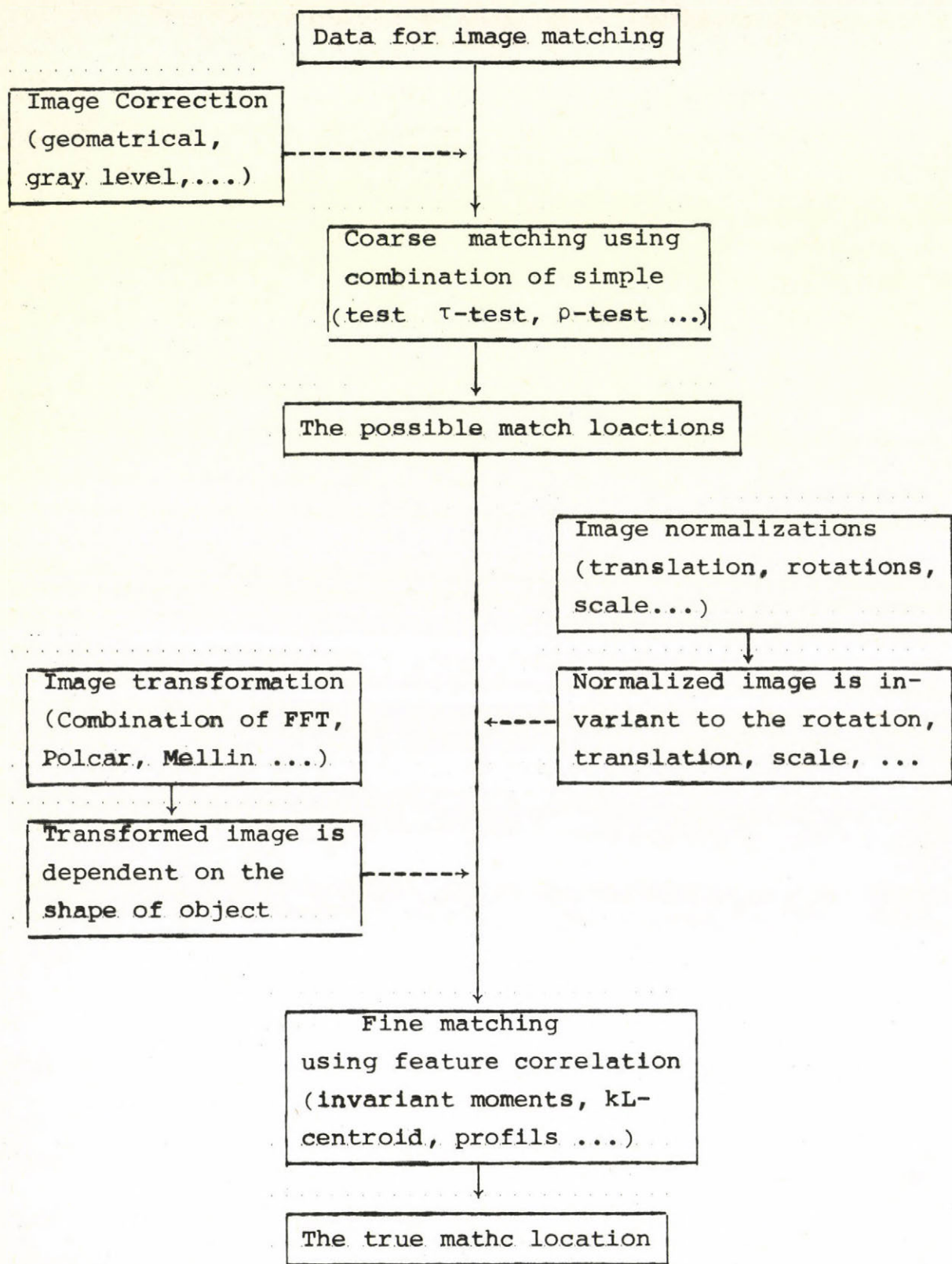


Fig.3. The scheme of synthesis of image matching algorithms



the low resolution level at which other methods, such as scene matching with edge features are not possible. Two improvements may be accomplished are the following:

- Weight each of the moments with an appropriate weighting factor before correlation.

- Generate higher-order moments. Select a set moments for correlation computation with the selection based on the information contents of the images.

At the same time, we also have used some other approaches as follows:

- The combination of transformations Fourier, Polcar, Mellin
- The image normalization in relation to certain transformations as translation, rotation, scale ...
- The kG-centroid (kL-centroid) features.

These methods have resulted in superior performance and were accomplished at greatly reduced computation and memory storage requirements.

REFERENCES

1. Y.Wong et al.: Scene matching with invariant moment. CGIP 8/1978, p. 16-24.
2. Y.Wong et al.: Sequential hierarchical scene matching. IEEE Trans on Comp. Vol C-27, 1978, p. 359-366.
3. G.Kanderburg: Coarse-fine template matching. SMC 7/1977. p. 104-107.
4. E.Hall: Computer image processing and recognition. Academic Press. New York 1979.
5. A. Rosenfeld et al.: Digital Image Processing. New York 1980.
6. A.Rosenfeld et al.: Relative effectiveness of selected texture primitive statistics for texture discrimination. SMC-11, 5/1981, p.360-370.
7. M.K.Hu: Visual pattern recognition by moment invariants. IRE Trans inform theory. Vol 67. 2/1962, p. 179-187.
8. S.Maitre: Moment Invariants. Proc. IEEE Vol 67, 4/1979. p. 697-699.
9. T.Hsia: A note on invariant moments in image processing. SMC 12/1981. p. 826-831.
10. N.Ahmed et al.: Orthogonal transforms for digital signal processing. New York 1975.
11. R.N.Bracewell: The Fourier transformation and its application New York 1965.
12. R.Chinins: A complet set of Fourier Descriptor for two-dimensional shapes. SMC-12, 1982, p. 848-855.
13. H.Arakwa: On-line recognition of hadwritten characters. PR. Vol 16, 1/1983, p.9-16.
14. D.Casasent et al.: New optical Transformation for Pattern Recognition. Proc. IEEE Vol 65, 1/1977, p.77-84.
15. R.Michael: Digital image shape detection. AFIF Vol 48 1979, p. 243-254.
16. V.Wedler: An unified approach for invariant properties and normalization of images. Rostock-Math-Kolloq-20, 1982. p.53-60.
17. C.H.Chien et al.: A normalized quadtree representation. CGIP 26, 1984, p. 331-346.

## APPENDIX I: A SET OF PROGRAMS FOR IMAGE MATCHING

Programs are written in FORTRAN-4 subroutines and were verified on PDP-11 minicomputer system. In the following subroutines

IMAG is an input image, a variable-array of size  $N \times N$   
I, J are coordinates of the upper-left corner of the windows

### 1. PROGRAM FOR GEOMETRICAL CORRECTION OF IMAGE

The call     CALL GECOR(IMAG, V, N, X, Y, XF, YF, K, MOD)

V : output image of size N N

X, Y the given points in IMAG

XF, YF : the given points in IMAG, which correspond to  
          X, Y in V

K : length of X, Y, XF, YF

MOD : correction mode

      MOD=0   correction by the way of 4-neighbor average

      MOD=1   correction by the way of nearest neighbor

### 2. PROGRAMS FOR COARSE MATCHING

#### 2.1 Computation of Histogram of a window

The call     CALL HISTO(IMAG, I, J, K, H, LEVEL)

K : size of window

LEVEL : gray level of image

H : output histogram of size LEVEL, in which H(i) is  
      number of pixels of gray level i+1.

#### 2.2 Computation of Profil of a window

The call     CALL PROFIL(IMAG, I, J, K, HX, HY)

K : size of windos

HX, HY : vectors of size K- profils on the x-axis and y-axis

### 2.3 Thresholding program

The call      CALL THRSLD(A,K,THR)

A : input data set of size K to be thresholded

THR : output threshold

## 3. PROGRAMS FOR NORMALIZATION OF IMAGE

### 3.1 Determination of the image center

The call      CALL IJCEN(IMAG,N,IC,JC)

IC,JC : output center of image

### 3.2 Centered transformation of image

The call      CALL TCEN(IMAG,CIMAG,N,M,IC,JC)

IC,JC : input coordinates of center

CIMAG : output image of size MxM

### 3.3 Determination of the rotation angle of image

The call      CALL TETA1(IMAG,N,T)

T : output rotation angle of image

### 3.4 Rotation of image

The call      CALL TROTA(IMAG,RIMAG,N,T,IT,JT)

RIMAG : output rotated image

T,IT,JT : given rotation angle and center

### 3.5 Scale of image

The call      CALL SCALE $\emptyset$ (IMAG $\emptyset$ ,IMAG,SIMAG,N,IC,JC)

IMAG,IMAG $\emptyset$  : input images of size NxN centered at IC,JC

SIMAG : output scaling image

## 4. PROGRAMS FOR TRANSFORMATIONS OF IMAGE

### 4.1 Polcar Transform

The call      CALL TPOCAR(IMAG,IA,N)

IA : output image of size NxN modified by Polcar transform

#### 4.2 Mellin transform

The call       MELIN(IMAG,FA,N,M)

FA : output image of size MxM modified by Mellin transform

#### 4.3 Fourier transform

The call       CALL TFØ(IMAG,FA,N)

FA : output image of xize NxN amplitude spectrum image

### 5. PROGRAMS FOR FINE MATCHING

#### 5.1 Computation of invariant moments of window

The call       CALL MOMENT(IMAG,K,I,J,KSI)

K : size of window

KSI : output vecotr consisting of loga of 7 invariant moments

#### 5.2 Computation of kL-centroid of window

The call       CALL TOPO(IMAG,I,J,CENTER,GRLV)

GRLV : gray level of image

CENTER 2, GRLV : output array

CENTER(1,1), CENTER(2,1) is the global center of image

CENTER(1,K), CENTER(2,K) are centers corresponding to the region of gray level K, K=1,...GRLV-1.

A kép-összeillesztő algoritmusok hatékony szintézise.

HOANG KIEM, PHAM NGOC KHOI

Összefoglaló

A szerző a kép-összeillesztő algoritmusok szintézisének hatékonyságára hierarchikus sémákat és "durva-finom" összeillesztő algoritmusokat használ. Tárgyalja a kép egyes tartományaira vonatkozó jellemző vonások kivonata-módszert, valamint a szintér-összeillesztő módszereket.

Эффективный синтез алгоритмов прикладывания образов

Хоанг Киём, Пхам Кхой

Резюме

Для повышения эффективности синтеза алгоритмов прикладывания образов авторы используют иерархические схемы и алгоритмы типа "грубо-тонкие". В статье дискитируются также некоторые другие методы.

## EXPERIENCES IN THE USE OF TWO DATABASE MANAGEMENT SYSTEMS FOR MICROCOMPUTERS

E. MUNIZ, M. FONFRIA, JORGE DE LA CANTERA

Central Research Institute for Computers  
CUBA

### INTRODUCTION

In the past few years, and as a result of the fast development in the microcomputer field and its introduction in almost all software areas, the general concept of database management systems (DBMS's) has suffered a little change. Until midseventies, database systems were conceived only for mainframes with fast I/O peripherals and large amounts of secondary storage (usually disk). This fact was dictated by the relative large resource requirements of such systems, given primarily for their ability to provide: a) high level of program and data independence, b) flexibility in the representation of data, c) high performance and efficiency in transaction-processing, d) suppression of data redundancy, e) capability of searching through data according to its attributes, and f) procedures for data security and recovery, among others. In practice, only a few DBMS implementations include all of these features, however, some have come close.

While microcomputers are being introduced and gaining popularity because of their interactive and simpler operation, increased the need for systems capable of integrate and compact rationally the currently available information. In this way, say "from bottom to the top" were approached the development of new systems featuring characteristics of DBMS's. In 1981, we could evaluate more than 10 of these systems.

However, even present-day DBMS's for microcomputers can not accomplish all of the objectives of a formal DBMS, although they are close according to the hardware currently available and offer the user a powerful tool for data processing.

In general, these systems provide:

- an easy way of creating screens and database reports,
- facilities for data entry and its relationing with all other data,
- provisions for program and data independence,
- facilities for using data for many applications,
- tools for creating applications with minimum programming effort, leaving every time on the user free for concentrate on his logical design.

Inmersed in this context, the authors have had the opportunity of developing an application intended to be used as a management aid in the manufacturing of some equipmnet, and its implementation under two DBMS's for micros: dBASE II from Ashton-Tate and Sensible-Solution from O'Hanlon Computer Systems.

In this paper, we describe the application, the general features of both database systems, and offer the conclusions of this realization, with the hope that they can help in giving some criterion about the use of these systems.

#### GENERAL FEATURES OF THE SYSTEM USED IN THE APPLICATION dBASE II

dBASE II is a powerful tool for database management that allow very easy handling of small to medium-scale databases using English-like commands. The system has the following major features:

- complete creation of databases
- facilities for adding, deleting, editing, displaying and printing database information
- data and program independence, i.e., changes to the program do not imply changes to the data and viceversa
- report generation from one or more databases
- use of the facilities of the terminal for editing data

dBASE II is typically interactive, having facilities for immediate correction of errors. A dBASE II program is a series of commands stored on a disk file which the user can execute by



means of the DO command. These command files do not need be previously compiled; commands are interpreted and immediately executed.

dBASE II allows the handling of relations (files) and memory variables. dBASE II files consist of a file header containing field descriptions, attributes and data all in compacted ASCII mode. The handling of files is accomplished in a manner transparent to the user. For example, for using a variable named COUNTER zero-initialized it is sufficient to utilize the command STORE 0 TO COUNTER without the need to declare or define COUNTER previously. The type of COUNTER (N: numeric, C: character, L: logical) is determined by the type of the source information, in this case numeric. Storing "ABC" to COUNTER will change its type to character.

If the user wants to modify the structure of a data file without losing information, he can easily utilize a suitable combination of COPY, MODIFY STRUCTURE and APPEND commands without the need to modify the command files that reference the restructured file. An important feature of dBASE II is the ease of learning by non-specialized people, the syntax of commands is very close to the human (English) language and also the documentation is clear and well presented.

dBASE II included a program named ZIP that allows the use of screens for data entry and also for displaying information. ZIP generates sequences of dBASE II commands that when executed perform the functions desired. Even though screen handling is not completely automatic in the sense that it has to be done by means of an external component, it has the advantage that the generated program can be edited and modified according to the user's needs.

In dBASE II direct access to relations' attributes is accomplished through dense index files, one for each attribute required by the user. When "opening" the relation the index file is specified and searches are carried out by FIND commands over the index field associated to the index file. If it is necessary later to change the search for another index field, the corresponding index file must be opened. The search command

(FIND) is very little flexible and do not allow automatic processing of records with duplicate keys. Reports can be obtained by the REPORT command, which can generate a program ready for execution. This command simplifies the process of report generation without the need for programming, though these facilities are of low level.

dBASE II shows up some inconveniences when relationing one file to another. This has to be done by means of the JOIN command, which outputs a file that in many cases is not completely necessary, leading to waste time and disk memory.

dBASE II allows processing of data files generated by other processors i.e., BASIC, FORTRAN, PASCAL and can produce files compatible with these processors. Also, recent versions include means for calling programs segments written in machine code, though this feature is very specific and aside from the general framework of users.

One limitation present in dBASE II is that only two files can be opened simultaneously, restricting the development of medium to large-scale professional applications. This limitation has been overcome in later releases of the systems.

dBASE II attains an efficient use of the hardware resources, requiring little memory. Also, it is necessary to point out that along with the simplicity of its language and the documentation it is easy to install it.

#### SENSIBLE-SOLUTION

This system shows up general features similar to that of dBASE II, however, his implementation differs in many aspects.

Sensible-Solution groups conceptually its functions into "tasks" that can be requested via a functions menu. The most important are:

- execution of command files
- data dictionary maintenance
- creation of screens
- command editing
- compile command files written in Sensible-Solution

- creation of programs
- database queries
- report generation
- restructuring files

Command files or language statements are not processed interactively, rather, they have to be previously compiled.

In Sensible-Solution all of the data including auxiliary variables must pertain to files controlled by the data dictionary (files RECFLE. MS, RECFLE.KS, FLDFLE.MS and FLDFLE.KS). Database information is structured around files comprised physically of two files: one contains user data in compacted ASCII mode called Master Data File, and the other contains a binary tree of pointers to records in the Master Data File; this file is called index-key-file and do always exist from the declaration of the file. This index file allows indexing a file up to 9 index fields and facilitate direct searching of attributes by the FIND statement, which can be indistinctively performed over any of the index fields. The various formats of the FIND statement are powerful tools for processing relations based upon its attributes, and the handling of two related files is performed easily by one of its variants (31 FND.REL.RC, Find Related Record).

Data definition is rigid and not very transparent to the user, and is always controlled by the data dictionary. For example, for using an auxiliary variable the user must invoke the task that updates the data dictionary and define the variable in an auxiliary file together with its declaration, and initialize that file. Similarly, for modifying the structure of a file without losing information, for example, modifying the length of a field, it is necessary to make changes in the data dictionary, reorganize the file and possibly reindexing it. Also, the changes are not "transparent" for the associated programs since all programs referencing the restructured file must be recompiled.

One powerful tool offered by Sensible-Solution is the definition and handling of screens. Screen definition includes labels

and windows to fields of a file that together with command files controls allow data entry, update and retrieve information from the specified file.

Sensible-Solution language is very peculiar and seems to be distant from the usual syntactical structures, that has two major disadvantages: one is the difficulty of learning by non-specialized users and the other is the necessity of a special task for editing command files, since the operating system's editor can not be used.

Sensible-Solution offers good tools for report generation. Through the definition of the report formats sophisticated reports can be obtained. This procedure adds great flexibility even though stands for the need of programming the report.

Sensible-Solution does not achieve automatic compatibility with other processors. One important feature of Sensible-Solution is its ability to open simultaneously up to 10 files, which is a great advantage when developing complex applications.

Sensible-Solution requires a large amount of primary and secondary storage, its components need to be distributed within 2 floppy disks each with at least 300K of free space, representing a constraint for its use on systems with 8 inch single density drives. Also, the installation procedure is not simple and requires a predetermined allocation of components within the two floppy disks.

#### DESCRIPTION OF THE APPLICATION

It is evident that all manufacturing processes involve the control of the different parts and elements that form the product, for example, it is necessary to know the list of elements that comprise every part of the product, the composition of the different models to be produced, and the stock of elements to carry out the process. The manual control of these aspects is a tedious and complex task, requiring generally one or more persons completely dedicated to this activity. For this reason, it was decided the implementation of a programming system on a microcomputer intended to provide an automated control of the

necessary operations supporting the manufacturing process, such as inventory control, allowing additionally that people unfamiliar with microcomputers could use the system, saving human and monetary resources, and obtaining reports with a high degree of reliability and in a small response time.

The most important functions offered by the system are:

- list elements or parts involved in the whole manufacturing process,
- list the composition of every part that constitutes the device,
- checking for the possibility of produce a specified number of parts depending upon current stock of elements printing deficits,
- extraction of components necessary for manufacturing a specified quantity of the device or some parts,
- listings that facilitate contracting elements for manufacturing a specified quantity of the device printing suppliers and prices,
- automation of the extraction and reception of components.

The programming system was designed to be supported by a database management system.

All the information handled by the system is included in the relations named PART, STOCK, TYPE, TRANSAC, and MODEL1, MODEL2, etc. Following is the description of the relations.

Relation PART

ORDER	ELEMENT	EQUIV1	EQUIV2	EQUIV3	PART:CODE	QUANTITY
-------	---------	--------	--------	--------	-----------	----------

ATTRIBUTE

DOMAIN

ORDER

numeric values of the form xx.yy, where xx identifies a particular type of element and yy is a consecutive

ELEMENT

the name of all elements used in the production of the device (all models)

EQUIV1 up to 3 equivalents to the main element, these fields can be blanks

EQUIV2 in case the equivalents do not exist.

EQUIV3 The inclusion of these attributes adds more flexibility in the extraction of elements, since the main element and its equivalents are handled indistinctively, when the stock of a component is exhausted

PART:CODE code of all parts comprising the device. Every part is formed by one or more elements.

QUANTITY numeric values indicating how many elements are used in a part.

This relation is interpreted as follows: ELEMENT (or EQUIV1, EQUIV2, EQUIV3) of type xx is used in PART:CODE in quantity QUANTITY.

Relation STOCK

ORDER	ELEMENT	MANUFACTURER	CODE	PRICE	QUANTITY:ON:HAND
-------	---------	--------------	------	-------	------------------

RECEPTIONS	EXTRACTIONS	DELIVERY
------------	-------------	----------

ATTRIBUTE

DOMAIN

ORDER the same values as the corresponding in the relation PART.

The objective of this field in this relation is to help in the control of the completeness and consistency of data and to facilitate its updating

ELEMENT the same values as ELEMENT, EQUIV1, etc., in the relation PART.

The ORDER of an element and its equivalents is the same.

MANUFACTURER	name of the manufacturer of each element
CODE	codes for identifying elements in the warehouse
PRICE	unit prices of the elements
QUANTITY: ON:HAND	stock of elements
RECEPTIONS	the sum of all receptions of each element
EXTRACTIONS	the sum of all extractions of each elements
SHIPPING	quantities extracted of each element to be printed in promissory notes. These values are retained until the user specifies another function involving extraction of components; this allows to obtain as many copies of promissory notes as desired and also in case of a system failures while printing, to repeat the procedure.

The relation STOCK is interpreted as the stock of all elements in the warehouse and the transactions in which each element is involved.

Relation TYPE

CONSEC	TITLE
--------	-------

ATTRIBUTE

DOMAIN

CONSEC

the values xx of the attribute ORDER of the relation PART

TITLE

description of the category of component represented by CONSEC

For example, the tuple (02, cables) can be interpreted as that the elements of type 2 are cables. This title is used in the listings obtained from the system's functions.

Generally, database management systems for microcomputers do not offer facilities for automatic recovery of information in case of failures, for this reason it was designed a relation named TRANSAC that keeps a history of transactions performed (receptions and extractions), ensuring more data security.

Relation TRANSAC

SPECIFICATION	QUANTITY	DATE	TYPE
---------------	----------	------	------

ATTRIBUTE

DOMAIN

SPECIFICATION	element or part handled in a transaction
QUANTITY	amount of elements or parts handled in the transaction
DATE	date when the transaction was performed
TYPE	the type of the transaction:(e)xtraction or (r)eception

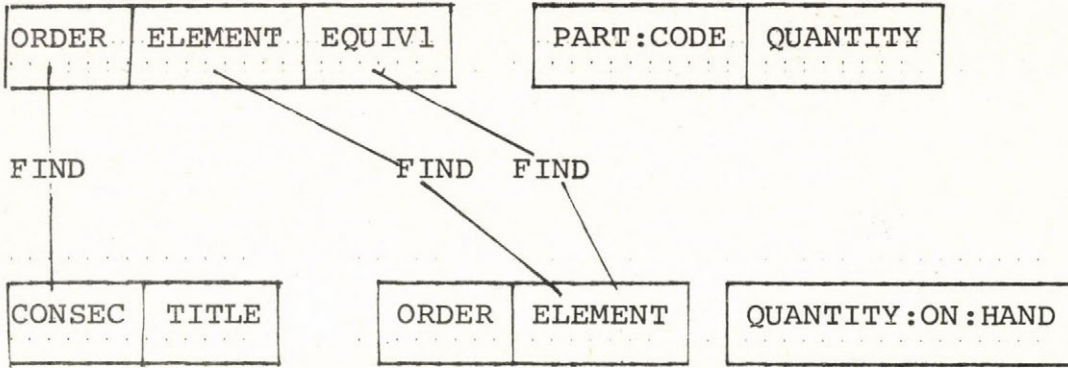
There are other relations named MODEL1, MODEL2,... MODELn, that keeps the composition of each of the different models.

Relation MODELJ

PART:CODE
-----------

The following diagram shows operations that are performed among relations in the major part of the system's functions.





Another operation that is performed when handling a model of the device is the following:

(description modelj)

obtaining the subset of description that comprises the modelj

## CONCLUSIONS

In general, Sensible-Solution performed the generality of the functions of the application faster than dBASE II due particularly to the command files have been already compiled, all related files can be opened simultaneously and the power and diversity of the search statements.

About Sensible-Solution, many of the functions could be integrated given the possibilities of screens and command file controls.

The programming and debugging resulted easier in dBASE II, particularly taking into account that a simple modification in the length of a field causes in Sensible-Solution modifications to the data dictionary, restructuring the file, etc. By the other hand, dBASE II's programming language is more mnemonic, consequently, it is learned easier and faster.

For simple applications handling a few files, dBASE II is a good choice. If the number of relations involved increases and should perform frequent and/or complex operations on databases, dBASE II can be ineffective, being advisable in this case the implementation of the application using Sensible-Solution.

REFERENCES

1. J.MARTIN, "Computer Database Organization".
2. C.J. DATE, "An introduction to Database Systems."
3. E. MUNIZ, and J. de la CANTERA "Design of a Control System under dBASE II". (to appear).
4. K.S.BARLEY, and J.R.DRISCOLL "A Survey of Database Management Systems for Microcomputers", BYTE, Nov 1981.
5. A.S.MICHAELS, B.MITTMAN, and C.R. CARLSON "A comparison of the Relational and CODASYL approaches to Database Management", Computer Surveys, Vol. 8, No. 1, March 1976.
6. J.DEMETROVICS, L.HANNAK and L.RONYAI "On functionally completeness of prime-element algebras", MTA-SZTAKI, Közlemények 25/1982.
7. M.W.BLASSEN, and K.P.ESWARAN "Storage and access in relational databases" IBM Systems Journal Vol. 16 num. 4/1977.
8. E.F.CODD, "A Relational Model of DATA for Large Shared Data Banks", Communications of the ACM Vol. 13/num. 6/June 1970.
9. E. LOWENTHAL, "Database systems for local nets", Datamation, august/1982 Vol. 28 num.9.
10. G.FLOAM, "Putting a Database on a mini", Datamation Vol. 22 num. 6. June 1976.

Tapasztalatok két adatbázis kezelő rendszer használatáról  
mikroszámítógépeken

E. MUNIZ, M. FONFRIA, J. DE LA CANTERA

Összefoglaló

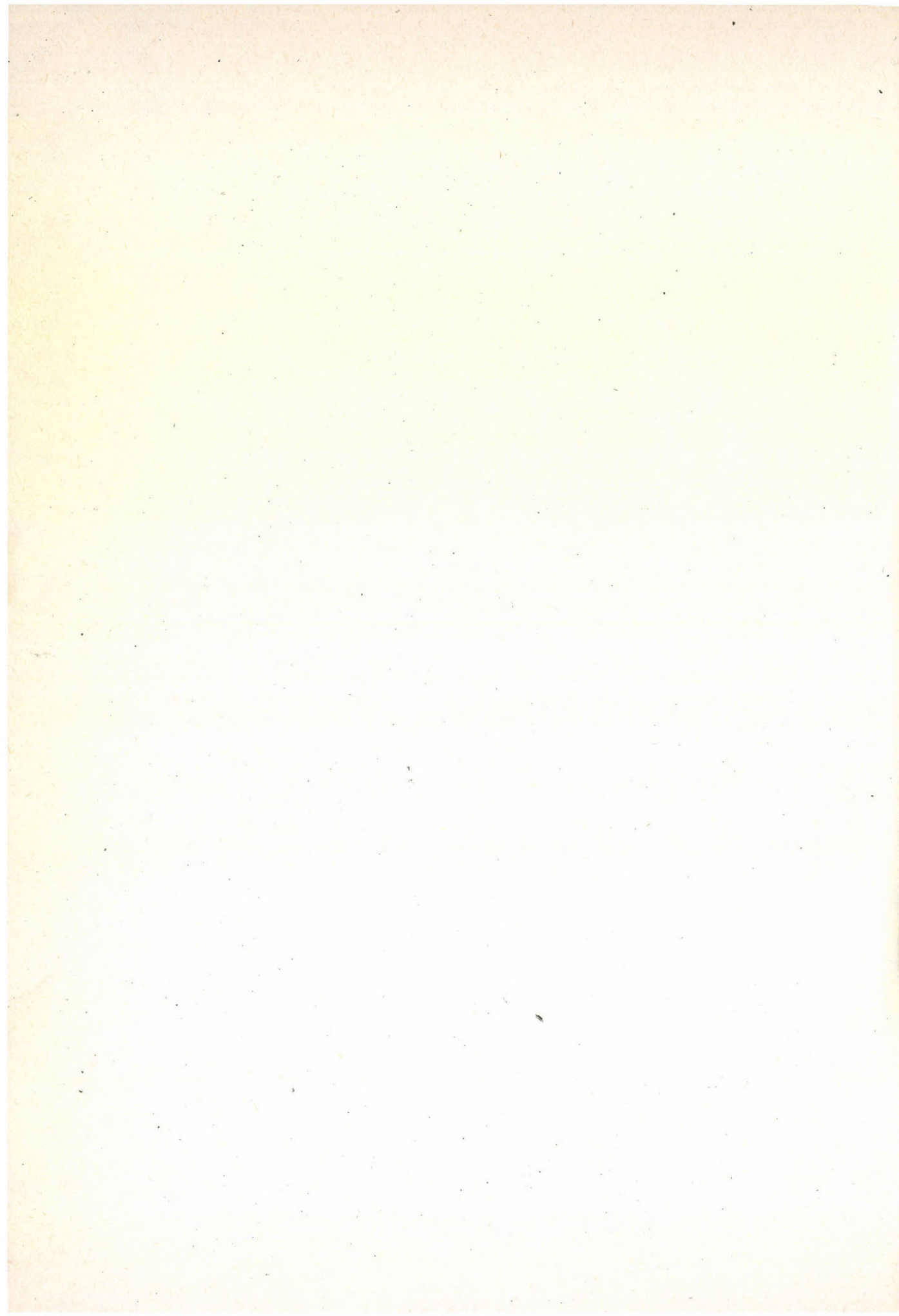
A szerzők az Ashton-Tate-féle dBASE II és az O'Hanlo Computer Systems-féle Sensible-Solution adatbáziskezelő rendszerekkel kapcsolatos tapasztalataikról számolnak be.

ОПЫТЫ ИСПОЛЬЗОВАНИЯ ДВУХ СИСТЕМ ОБРАБОТКИ ДАННЫХ НА МИКРО-ЭВМ

Е. Миниз, М. Фонфрия, Й. Де Ла Кантера

Резюме

Авторы описывают опыты проведенные с системами dBASE II /Ashton-Tate/ и Sensible-Solution /O'Hanlo Computer Systems/.



ON THE  $c$ -SEPARABLE AND DOMINANT SETS  
OF VARIABLES FOR THE FUNCTIONS

S. SHTRAKOV

2700 Blagoevgrad  
School of Education of Blagoevgrad  
Bulgaria

In this paper we investigate some properties of the  $c$ -separable and dominant sets which are introduced immediately. We use some notations and terminology from [1,2,3].

Let  $f$  be a function,  $R_f$  - the set of all essential variables for  $f$  and  $S_f$  - the set of all separable sets of  $f$ .

*Definition 1.* A set  $M$ ,  $M \subseteq R_f$  is called  $c$ -separable for  $f$  with respect to  $N = \{x_{i_1}, x_{i_2}, \dots, x_{i_s}\} \subseteq R_f$ , if for every  $s$ -values  $c_{i_1}, c_{i_2}, \dots, c_{i_s}$  of the variables in  $N$ , the subfunction of  $f$  which is obtained with these values, depends on all variables of  $M$  i.e.  $M \subseteq R_f(x_{i_1}=c_{i_1}, x_{i_2}=c_{i_2}, \dots, x_{i_s}=c_{i_s})$

When  $M$  is a  $c$ -separable set for  $f$  with respect to  $R_f \setminus M$ , it is called  $c$ -separable for  $f$ . The set of all  $c$ -separable sets for  $f$  with respect to  $N$  will be denoted by  $S_{f,N}^*$  and  $S_f^* = \{K | K \in S_{f,R_f \setminus K}^*\}$ .

*Definition 2.* A set  $M = \{x_{i_1}, x_{i_2}, \dots, x_{i_m}\} \subseteq R_f$ , is called dominant set over the set  $N$ ,  $N \subseteq R_f$  for  $f$ , if there exist  $m$  - values  $c_{i_1}, c_{i_2}, \dots, c_{i_m}$  of the variables in  $M$  such that

$$N \cap R_f(x_{i_1} = c_{i_1}, x_{i_2} = c_{i_2}, \dots, x_{i_m} = c_{i_m}) = \emptyset$$

and  $M$  is a minimal set with respect to this property.

When this equation is true and  $M$  is a dominant set over  $N$ , it is said that  $M$  dominates over  $N$  with the values  $c_{i_1}, c_{i_2}, \dots, c_{i_m}$ .

The set of all dominant sets over  $N$  will be denoted by  $L_{N,f}$  and  $D_{N,f} = \{x_\alpha \in R_f \mid (\exists M) x_\alpha \in M \wedge M \in L_{N,f}\}$ .

The proofs of the next lemmas follow immediately from the Definitions 1 and 2,

Lemma 1. If  $M_i \in S_{f, N}^*$ ,  $i \in I$ , then  $\bigcup_{i \in I} M_i \in S_{f, N}^*$ .

Lemma 2. If  $M \in S_{f, N}^*$  then for every  $N_1, N_1 \subseteq N$  the set  $M$  belongs to  $S_{f, N_1}^*$ .

Lemma 3. If  $M \in S_{f, N}^*$  then for every  $M_1, M_1 \subseteq M$  the set  $M_1$  belongs to  $S_{f, N}^*$ .

Lemma 4. Let  $M \subseteq R_f$  and  $N = \{x_{j_1}, x_{j_2}, \dots, x_{j_s}\} \subseteq R_f$ . If there exist the values  $c_{j_1}, c_{j_2}, \dots, c_{j_s}$  such that

$$M \cap R_f(x_{j_1} = c_{j_1}, x_{j_2} = c_{j_2}, \dots, x_{j_s} = c_{j_s}) = \emptyset$$

then there is a subset  $N_1$  of  $N$  such that  $N_1 \in L_{M,f}$ .

Theorem 5. If  $M \in L_{N,f}$ ,  $N \in L_{P,f}$  and  $M \cap N = \emptyset$  then there exists  $M_1$  such that  $M_1 \subseteq M$  and  $M_1 \in L_{P,f}$ .

Proof. We can suppose without loss of generality that  $M = \{x_1, x_2, \dots, x_m\}$  and  $N = \{x_{i_1}, x_{i_2}, \dots, x_{i_s}\}$ .

Let  $c_1, c_2, \dots, c_m$  and  $c_{i_1}, c_{i_2}, \dots, c_{i_s}$  be some values which  $M$  dominates over  $N$  and  $N$  dominates over  $P$ . If

$$f_1 = f(x_1 = c_1, x_2 = c_2, \dots, x_m = c_m)$$

then for every  $s$ -values  $\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_s}$  of the variables in  $N$  we obtain

$$f_1 = f_1(x_{i_1} = \alpha_{i_1}, x_{i_2} = \alpha_{i_2}, \dots, x_{i_s} = \alpha_{i_s}).$$

Hence

$$f_1 = f_1(x_{i_1} = c_{i_1}, x_{i_2} = c_{i_2}, \dots, x_{i_s} = c_{i_s})$$

and by  $M \cap N = \emptyset$  it follows

$$f_1 = f(x_1=c_1, x_2=c_2, \dots, x_m=c_m, x_{i_1}=c_{i_1}, x_{i_2}=c_{i_2}, \dots, x_{i_s}=c_{i_s})$$

and  $P \cap Rf_1 = \emptyset$ . By Lemma 4 it follows that there is  $M_1$  such that  $M_1 \subseteq M$  and  $M_1 \in L_{p,f}$ .

The condition  $M \cap N = \emptyset$  is essential which may be seen from the following example. Let

$$f = x_1^0 x_3 + x_2 x_4 x_5 + x_1 x_4 x_5 \pmod{3}$$

where 
$$x_1^0 = \begin{cases} 1 & \text{if } x_1 = 0 \\ 0 & \text{if } x_1 \neq 0 \end{cases}$$

If  $M = \{x_1, x_2\}$ ,  $N = \{x_1, x_4\}$  and  $P = \{x_3, x_5\}$  then  $M \in L_{N,f}$ ,  $N \in L_{p,f}$  but there isn't any set  $M_1$  such that  $M_1 \subseteq M$  and  $M_1 \in L_{p,f}$ .

*Lemma 6.* For every  $x_\alpha, x_\alpha \in R_f$ , the set  $\{x_\alpha\}$  belongs to  $L_{\{x_\alpha\},f}$ .

*Proof.* For every value  $c_\alpha$  of the variable  $x_\alpha$  it holds true  $\{x_\alpha\} \cap R_f(x_\alpha = c_\alpha) = \emptyset$ .

But  $\{x_\alpha\}$  hasn't any nonempty proper subset and by Theorem 5 it follows  $\{x_\alpha\} \in L_{\{x_\alpha\},f}$ .

*Theorem 7.* If  $x_\alpha \in R_f$  and  $x_\beta \in D_{\{x_\alpha\},f}$  then  $\{x_\alpha, x_\beta\} \in S_f$ .

*Proof.* We can suppose without loss of generality that

$$M = \{x_\beta, x_3, x_4, \dots, x_m\} \in L_{\{x_\alpha\},f}$$

If  $x_\alpha = x_\beta$  then the theorem is trivial. Now, let  $x_\alpha \neq x_\beta$ . If we suppose that  $x_\alpha \in M$  then by Lemma 6 it follows  $M \notin L_{\{x_\alpha\},f}$ . It is a contradiction. Hence  $x_\alpha \notin M$ .

Let  $c_\beta, c_3, c_4, \dots, c_m, c_{m+1}, \dots, c_n$  be  $n-1$ -values of the variables in  $R_f \setminus \{x_\alpha\}$ ,  $|R_f| = n$ , such that  $x_\alpha \in Rf_1$  and  $x_\alpha \notin Rf_2$ , where

$$f_1 = f(x_3 = c_3, x_4 = c_4, \dots, x_m = c_m, x_{m+1} = c_{m+1}, \dots, x_n = c_n)$$

and

$$f_2 = f(x_\beta = c_\beta, x_3 = c_3, x_4 = c_4, \dots, x_m = c_m).$$

This choice of  $c_\beta, c_3, c_4, \dots, c_m, c_{m+1}, \dots, c_n$  is possible because  $x_\alpha \in R_f$  and  $M \in L_{\{x_\alpha\}, f}$ . On the supposition that  $\{x_\alpha, x_\beta\} \notin S_f$  we obtain  $f_1 = f_1(x_\beta = c'_\beta)$  for every  $c'_\beta$ .

In particular when  $c'_\beta = c_\beta$  it follows  $x_\alpha \in R_{f_1}(x_\beta = c_\beta)$  i.e.  $x_\alpha \in R_{f_2}$ . This a contradiction. The theorem is proved.

*Corollary.* If  $x_\alpha \in D_{\{x_\beta\}, f}$  or  $x_\beta \in D_{\{x_\alpha\}, f}$  then  $\{x_\alpha, x_\beta\} \in S_f$ .

*Theorem 8.* If  $M \in L_{N, f}$  and there is a value  $c_\alpha$  of the variable  $x_\alpha$  such that  $M \not\subseteq R_f(x_\alpha = c_\alpha)$  then  $x_\alpha \in D_{N, f}$ .

*Proof.* Let  $M = \{x_1, x_2, \dots, x_m\}$ . If  $x_\alpha \in M$  then the theorem is trivial. Now, let  $x_\alpha \notin M$  and  $c_\alpha$  be a value of the variable  $x_\alpha$  such that  $M \cap R_{f_1} \neq M$ , where  $f_1 = f(x_\alpha = c_\alpha)$ . We can suppose without loss of generality that  $x_1 \notin R_{f_1}$ . Let  $c_1, c_2, \dots, c_m$  be  $m$  values of the variables in  $M$  such that

$$N \cap R_f(x_1 = c_1, x_2 = c_2, \dots, x_m = c_m) = \emptyset$$

Then for every  $m$  - values  $c'_1, c'_2, \dots, c'_m$  of the variables  $x_1, x_2, \dots, x_m$  it holds true

$$N \cap R_f(x_2 = c'_2, x_3 = c'_3, \dots, x_m = c'_m) \neq \emptyset \text{ and } f_1 = f_1(x_1 = c'_1).$$

This equation implies

$$N \cap R_f(x_\alpha = c_\alpha, x_2 = c_2, \dots, x_m = c_m) = \emptyset$$

By Lemma 4 there is a subset  $M_1$  of  $M'$  such that  $M_1 \in L_{N, f}$  where  $M' = \{x_\alpha, x_2, x_3, \dots, x_m\}$ .

Now, if  $x_\alpha \notin M_1$  then  $M \notin L_{N, f}$ . This is a contradiction.



Hence  $x_\alpha \in M_1$ . The theorem is proved.

*Corollary.* For every essential variable of the function  $f$ ,  $\{x_\alpha\} \in S_f^*$  if and only if  $D_{\{x_\alpha\}, f} = \{x_\alpha\}$ .

*Theorem 9.* For every  $N$ ,  $N \subseteq R_f$  the set  $D_{N, f}$  is a  $c$ -separable set for  $f$ .

*Proof.* If  $N = \emptyset$  then obviously  $D_{N, f} \in S_f^*$ . Now, let  $N \neq \emptyset$  and we can suppose without loss of generality that  $R_f = \{x_1, x_2, \dots, x_n\}$  and  $D_{N, f} = \{x_1, x_2, \dots, x_p\}$ ,  $p \leq n$ . Moreover, we suppose that there are  $n-p$  - values  $c_{p+1}, c_{p+2}, \dots, c_n$  of the variables in  $R_f \setminus D_{N, f}$  such that  $D_{N, f} \not\subseteq R_{f_1}$ , where

$$f_1 = f(x_{p+1} = c_{p+1}, x_{p+2} = c_{p+2}, \dots, x_n = c_n).$$

Again, we can suppose without loss of generality that  $x_p \notin R_{f_1}$  and

$$M = \{x_p, x_{i_2}, x_{i_3}, \dots, x_{i_m}\} \in L_{N, f}.$$

Then for every  $m-1$  - values  $c_{i_2}, c_{i_3}, \dots, c_{i_m}$  of the variables in  $M \setminus \{x_p\}$  it holds true

$$N \cap R_f(x_{i_2} = c_{i_2}, x_{i_3} = c_{i_3}, \dots, x_{i_m} = c_{i_m}) \neq \emptyset.$$

Now, we suppose that there are the values  $c'_{i_2}, c'_{i_3}, \dots, c'_{i_m}, c'_{p+1}, \dots, c'_n$  such that  $N \cap R_{f_2} = \emptyset$  where

$$f_2 = f(x_{i_2} = c'_{i_2}, x_{i_3} = c'_{i_3}, \dots, x_{i_m} = c'_{i_m}, x_{p+1} = c'_{p+1}, \\ x_{p+2} = c'_{p+2}, \dots, x_n = c'_n).$$

By Lemma 4 there is a subset  $M_1$  of  $M'$  such that  $M_1 \in L_{N, f}$ , where  $M' = \{x_{i_2}, x_{i_3}, \dots, x_{i_m}, x_{p+1}, \dots, x_n\}$ . By  $M \in L_{N, f}$  we obtain  $M_1 \notin D_{N, f}$  which is a contradiction. Consequently, for every  $m+n-p-1$  -values  $\alpha_{i_2}, \alpha_{i_3}, \dots, \alpha_{i_m}, \alpha_{p+1}, \alpha_{p+2}, \dots, \alpha_n$

of the variables in  $(R_f|_{D_N, f}) \cup (M|\{x_p\})$  it holds true

$$N \cap R_f(x_{i_2} = \alpha_{i_2}, x_{i_3} = \alpha_{i_3}, \dots, x_{i_m} = \alpha_{i_m}, x_{p+1} = \alpha_{p+1}, x_{p+2} = \alpha_{p+2}, \dots, x_n = \alpha_n) \neq \emptyset.$$

But  $f_1 = f_1(x_p = \alpha_p)$  for every  $\alpha_p$  and there exist  $m$ -values  $c''_p, c''_{i_2}, c''_{i_3}, \dots, c''_{i_m}$  of the variables in  $M$  such that

$$N \cap R_f(x_p = c''_p, x_{i_2} = c''_{i_2}, x_{i_3} = c''_{i_3}, \dots, x_{i_m} = c''_{i_m}) = \emptyset$$

and

$$N \cap R_f(x_p = c''_p, x_{i_2} = c''_{i_2}, \dots, x_{i_m} = c''_{i_m}, x_{p+1} = c_{p+1}, \dots, x_n = c_n) = \emptyset$$

This is a contradiction. The theorem is proved.

The author is indebted to K.N. Cimev for his kind encouragement and advice.

## REFERENCES

- [1] K.N. Cimev, On some properties of functions. Colloq. Math. Soc. J. Bolyai, 28, Szeged, 1979.
- [2] K.N. Cimev, Separable sets of arguments of functions. Blagoevgrad, 1983. (in Bulgarian).
- [3] K.N. Cimev, Functions and graphs. Blagoevgrad, 1983. (in Bulgarian).

A függvények változóinak c-szeparábilis és domináns halmazairól.

S. SHTRAKOV

Összefoglaló

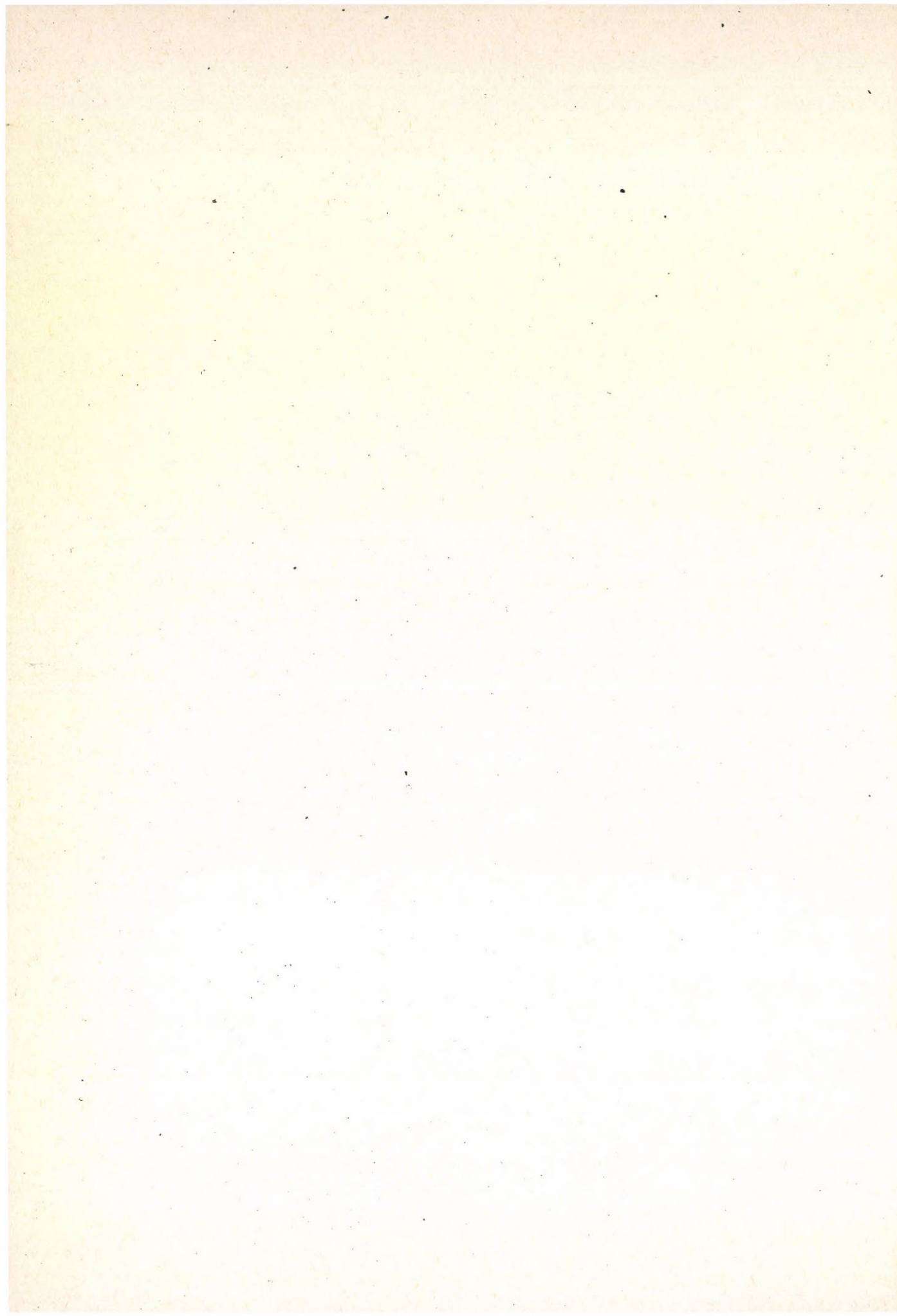
A szerző bevezeti a c-szeparábilis és domináns halmazok fogalmát és néhány ezen fogalmakat jellemző tételt bizonyít be.

Об c-сепарабельных и доминантных множествах переменных для функций

С. Штраков

Резюме

Автор дает определение c-сепарабельных и доминантных множеств и доказывает несколько теорем, которые характеризуют эти множества.



## SOME SPECIAL SPERNER-SYSTEMS

VU DUC THI

Computer and Automation Institute  
Hungarian Academy of Sciences

## §0. INTRODUCTION

One of the possible models of a data base is the relational model introduced by Codd [3]. The minimal keys play important roles for the logic and structural investigation of this model. In [5], it has been proved the equivalence of minimal keys with Sperner-systems.

In this paper we investigate some special Sperner-systems and some results of the functional dependencies.

## §1. DEFINITIONS

In this section, we present necessary definitions.

Definition 1.1. Let  $\Omega = \{1, \dots, n\}$  and  $P(\Omega)$  its power set. The function  $F: P(\Omega) \rightarrow P(\Omega)$  is called a closure operation or closure iff for every  $A, B \in P(\Omega)$

- (1)  $A \subseteq F(A)$
- (2)  $A \subseteq B \rightarrow F(A) \subseteq F(B)$
- (3)  $F(F(A)) = F(A)$

Definition 1.2. Let  $F$  be a closure operation over  $\Omega$  and  $A \subseteq \Omega$ .  $A$  is a key of  $F$  if  $F(A) = \Omega$ .

Definition 1.3. Let  $F$  be a closure operation over  $\Omega$ .  $A$  ( $A \subseteq \Omega$ ) is said to be a minimal key of  $F$  if  $A$  is key but  $F(B) \neq \Omega$  for any proper subset  $B$  of  $A$ . We denote

$$K_F = \{A: F(A) = \Omega, (\forall B \subseteq A)(F(B) = F(A) \rightarrow B = A)\}$$

It is clear that  $K_F$  is set of minimal keys.

Definition 1.4. Let  $\Omega = \{1, \dots, n\}$  and  $K \subseteq P(\Omega)$

$K$  is said to be a Sperner-system over  $\Omega$  if arbitrary  $A, B \in K$  then  $A \not\subseteq B$ .

It is easy to see that the set of minimal keys of an arbitrary closure operation create a Sperner-system.

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

Definition 1.5.

$$K^{-1} = \{A \subseteq \Omega: (B \in K \rightarrow B \not\subseteq A) \text{ and } (A \subset C) \rightarrow (\exists B \in K)(B \subseteq C)\}$$

It is clear that  $K^{-1}$  is family of the subsets of  $\Omega$  not containing the elements of  $K$  and which are maximal for this property. It is obvious that  $K^{-1}$  is also a Sperner-system.

## §2. SOME SPECIAL SPERNER-SYSTEMS

The minimal key is an important concept in the logic and structural investigation of relational datamodel.

The antikeys play important roles for investigation of the extremal problems of functional dependencies as well as for the construction of a concrete matrix representing a set of minimal keys or for finding minimal keys.

Remark 2.1. In [1,6] it has been proved one important result that if  $K$  is an arbitrary Sperner-system then there is a closure operation  $F$  ( $F'$  if  $\Omega \notin K$ ) for which  $K = K_F$  ( $K = K_{F'}^{-1}$ )

[7] defined a saturated Sperner-System, as follows:

A Sperner-system  $K$  is saturated if  $\{B\} \cup K$  is not Sperner-system for any  $B$ .

In [2,7], it has been proved that if  $K$  is a saturated Sperner-system then  $K=K_F$  uniquely determines  $F$ . Now, we investigate some special Sperner-systems which connected with a saturated Sperner-systems.

Definition 2.2. Let  $\Omega = \{1, \dots, n\}$  and  $K$  is a Sperner-system over  $\Omega$ .  $K$  is united if  $K \cup K^{-1}$  is Sperner-system.

It is clear that  $K$  is an united Sperner-system then  $K \cup K^{-1}$  is saturated.

EXAMPLE 2.3. Let  $\Omega = \{1, 2, 3, 4, 5, 6\}$  and

$K = \{(1, 2), (3, 4), (5, 6, 7)\}$ . It is clear that

$K^{-1} = \{(1, 3, 5, 6), (1, 3, 5, 7), (1, 3, 6, 7), (2, 3, 5, 6),$   
 $(2, 3, 5, 7), (2, 3, 6, 7), (1, 4, 5, 6), (1, 4, 6, 7),$   
 $(2, 4, 5, 6), (2, 4, 5, 7), (2, 4, 6, 7)\}$  and  $K$  is united.

Now, we define a closed set, as follows:

Let  $F$  be a closure operation over  $\Omega$  and  $A \subseteq \Omega$ .  $A$  is called closed in  $F$  if  $F(A) = A$ .

Denote  $Z(F) = \{A : F(A) = A\}$

$T(F) = \{A \subseteq \Omega : F(A) = A \text{ and } \exists B \in Z(F) \setminus \{\Omega\} : A \subset B\}$

That is:  $Z(F)$  is the family of closed sets and  $T(F)$  is the family of maximal closed sets.

Lemma 2.4. Let  $F$  be a closure operation over  $\Omega$ .

$$T(F) = K_F^{-1}.$$

Proof. Let  $A$  be a maximal closed set but there is  $B(B \in K_F)$  such that  $B \subseteq A$ , then  $F(A) = \Omega$ . This contradicts to  $A \subseteq \Omega$ .

If  $A \subset D$  (where  $D \subseteq \Omega$ ) then it is clear that  $F(D) = \Omega$  (because  $A \in T(F)$ ). Consequently  $A$  is an antikey.

Conversely, now assume that  $A$  is an antikey but  $A \subseteq F(A)$ .

Hence  $F(F(A)) = F(A) = \Omega$ . Consequently  $A$  is a key. This contradicts to  $\forall B \in K_F : B \not\subseteq A$ . Suppose that there is  $A \subset A'$  and  $A' \in Z(F) \setminus \{\Omega\}$ , then  $A'$  is a key. This a contradicts to  $A' \subseteq \Omega$ .

The lemma is proved.

Theorem 2.5. Let  $\Omega = \{1, \dots, n\}$  and  $K$  is a Sperner-system  $K$  is united if and only if for every  $A \in K$  there exists  $B(B\theta(K^{-1})^{-1})$  such that  $A \subseteq B$ . Where  $(K^{-1})^{-1}$  is set of the antikeys of  $K^{-1}$ .

Proof. Let  $K$  is an united Sperner-system and suppose that  $A \in K$ . By remark 2.1 for  $K^{-1}$  there exists a closure operation  $F$  such that  $K_F = K^{-1}$ . If  $F(A) = \Omega$  then there is  $C \in K^{-1}$  for which  $C \subseteq A$ . This contradicts to  $K$  is united. Hence  $F(A) \subset \Omega$ . Basing on lemma 2.4 it is known that a set of antikeys is family of the maximal closed sets. It is clear that  $F(A)$  is closed set. Consequently there exists  $B \in (K^{-1})^{-1}$  such that  $A \subseteq B$ .

On the other side if for every  $A \in K$  there is  $B \in (K^{-1})^{-1}$  such that  $A \subseteq B$ . Denote  $T = \{B \in (K^{-1})^{-1} : A \in K : A \subseteq B\}$

It can be seen that for the arbitrary Sperner-system  $H$  if  $D \in H$  then whether  $\{D\} \cup H^{-1}$  is a Sperner-system or there is  $C \in H^{-1}$  for which  $C \subset D$ . By  $A \subseteq B$  and according to the definition of  $K^{-1}$  then  $T \cup \{S\}$  is a Sperner-system, where  $S$  is the arbitrary element of  $K^{-1}$ . Consequently,  $K \cup \{S\}$  is Sperner-system. That is  $K \cup K^{-1}$  is Sperner-system. The theorem is proved.

It is clear that if  $K$  is an united Sperner-system then  $K^{-1}$  and  $(K^{-1})^{-1}$  are a non-saturated Sperner-systems.

Remark 2.6. There exists an example which show that  $K$  is united but  $K^{-1}$  isn't united.

Let  $\Omega = \{1, 2, 3, 4, 5\}$  and  $K = \{(1, 2, 3), (2, 3, 5)\}$

According to the algorithm [8] for finding the set of antikeys we have:

$$K^{-1} = \{(1, 2, 4, 5), (1, 3, 4, 5), (2, 3, 4)\} \text{ and}$$

$$(K^{-1})^{-1} = \{(1, 2, 3, 5), (1, 4, 5), (1, 4, 3), (3, 4, 5), (1, 2, 4), (2, 4, 5)\}$$

By  $\{1, 4, 5\} \in (K^{-1})^{-1}$  and  $\{1, 2, 4, 5\} \in K^{-1}$  then  $K$  is united but  $K^{-1}$  is not united.



Definition 2.7. Let  $K$  be a Sperner-system over  $\Omega$ .  $K$  is embedded if for every  $A \in K$  there is  $B \in H$  such that  $A \subset B$ . Where  $H^{-1} = K$ .  
We have:

EXAMPLE 2.8. Let  $\Omega = \{1, 2, 3, 4, 5, 6\}$  and  $N = \{(1, 2), (3, 4), (5, 6)\}$ .  
It is clear that  $N^{-1} = \{(1, 3, 5), (1, 3, 6), (1, 4, 5), (1, 4, 6), (2, 3, 5), (2, 3, 6), (2, 4, 5), (2, 4, 6)\}$ .  
Denote  $K = N \cup N^{-1}$ , it can be seen that  $K$  is saturated. We use the algorithm which find a set of antikeys. Then

$$K^{-1} = \{(1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 5), (3, 6), (4, 5), (4, 6)\}$$

By  $K^{-1}$   $\{1, 2\}$  is Sperner-system then  $K^{-1}$  isn't saturated.  
That is: There exists  $K$  is saturated and  $K^{-1}$  isn't saturated.  
But we have:

Theorem 2.9. Let  $K$  be a Sperner-system over  $\Omega$ .  
 $K$  is saturated if and only if  $K^{-1}$  is embedded.

Proof. Let  $K$  be a saturated Sperner-system and according to the definition of  $K^{-1}$ , it is clear that  $K^{-1}$  is embedded.

Conversely, if  $K^{-1}$  is a embedded Sperner-system, but  $K$  isn't saturated. Consequently, there exists  $A \subset \Omega$  such that  $K \cup \{A\}$  is Sperner-system. It is clear that for every  $C \in K$  then  $C \subset \Omega$ . Hence we can construct a  $B$  such that  $A \subset B$  and  $K \cup \{B\}$  is Sperner-system and for every  $B'$  ( $B \subset B'$ ) there is  $C \in K$  such that  $C \subset B'$ . It can be seen that  $B \in K^{-1}$ . This contradicts with  $K^{-1}$  is embedded. The theorem is proved.

Now, we define an inclusive Sperner-system, as follows:

Definition 2.10. Let  $K$  be a Sperner-system. We say that  $K$  is inclusive if for every  $A \in K$  there exists  $B \in K^{-1}$  such that  $B \subset A$ .

Theorem 2.11.

$K$  is an inclusive Sperner-system if and only if  $K^{-1}$  is saturated.

Proof. Now, assume that  $K$  is an inclusive Sperner-system but  $K^{-1}$  isn't saturated. By the definition of  $K^{-1}$  there is  $B \in (K^{-1})^{-1}$  such that  $K^{-1} \cup \{B\}$  is Sperner-system. By remark 2.1. for  $K$  there exists a closure operation  $F$  for which  $K_F = K$ . If  $F(B) \in \Omega$  then by lemma 2.4 there is  $A \in K^{-1}$  such that  $F(B) \subseteq A$  (because set of antikeys is family of the maximal closed sets). This contradicts to  $K^{-1} \cup \{B\}$  is Sperner-system. Consequently,  $B$  is a key. We use the algorithm [8] which find a minimal key then it can be seen that there exists  $B'$  ( $B' \subseteq B$ ) such that  $B' \in K$  and it is clear that  $K^{-1} \cup \{B'\}$  is Sperner-system. This contradicts with the definition of  $K$ . That is:  $K^{-1}$  is saturated. On the other side by the definition of  $K^{-1}$  and  $K^{-1}$  is saturated then it is clear that  $K$  is inclusive. The theorem is proved.

Proposition 2.12.

There exists Sperner-system  $K$  that (Denote  $H^{-1} = K$ )

- (1) or  $K$  is saturated, but  $K^{-1}$  isn't saturated.
- (2) or  $K$  is saturated, but  $H$  isn't saturated
- (3) or  $K$  is embedded, but  $K^{-1}$  isn't embedded
- (4) or  $K$  is embedded, but  $H$  isn't embedded
- (5) or  $K$  is inclusive, but  $K^{-1}$  isn't inclusive
- (6) or  $K$  is inclusive, but  $H$  isn't inclusive
- (7) or  $K$  is united, but  $K^{-1}$  isn't united
- (8) or  $K$  is united, but  $H$  isn't united.

Proof. By an example 2.8 we have (1). By the theorem 2.9  $(K^{-1})^{-1}$  isn't embedded in this example. Hence we have (3). By remark 2.6 we have (7). By theorem 2.11 in the example 2.8  $H$  is inclusive (where  $H^{-1} = K$ ). Now, suppose that if  $K$  is inclusive then set of antikeys of  $K$  is also inclusive. Consequently in the example 2.8  $H$  is inclusive the  $K$  is inclusive. By theorem 2.11  $K^{-1}$  is saturated. But in this example  $K^{-1}$  isn't saturated. Hence we have (5). For (2) we can prove as follows: Let  $\mathcal{K}$  be a Sperner-system. Denote  $K^1 = K$  and  $K^n$  is Sperner-system, a set of the antikeys of which is  $K^{n-1}$  (where  $n \geq 2$ ). We know that the number of the Sperner-systems over  $\Omega$  is finite (maximum is  $2^{2^{|\Omega|}}$ ). On the

other side  $K$  and  $K^{-1}$  are determined uniquely by each other. Consequently, there is the number  $m$  ( $2 \leq m \leq 2^{|\Omega|}$ ) such that  $K^m = K$  and  $K^{m-1} = K^{-1}$ . If suppose that  $K$  is saturated then  $H$  is also saturated (where  $H^{-1} = K$ ). Then this means that  $K^p$  ( $2 \leq p \leq m$ ) is also saturated. Consequently  $K^{-1}$  is saturated. This contradicts with the example 2.8. That is: there exists a Sperner-system  $K$  such that  $K$  is saturated, but  $H$  is not saturated.

By similar arguments we have also (4), (6) and (8).

The proposition is proved.

Let  $K_1$  and  $K_2$  are Sperner-systems. We say that the union  $K = K_1 \cup K_2$  is proper saturated if  $A \in K_1$  then  $A \notin K_2$  and  $K$  is saturated.

It is clear that if  $K$  is united then  $K \cup K^{-1}$  is proper saturated.

Proposition 2.13. Let  $\Omega = 1, \dots, n$  and  $K$  is arbitrary. Sperner-system. Then  $K \cup (K^{-1})^{-1}$  is not proper saturated.

Proof. It is clear that  $K^{-1}$  is Sperner-system. Now, we investigate the first case:  $K^{-1}$  is saturated. Consequently, by theorem 2.11 we denote  $T_1 = \{B \in K^{-1} : \exists A \in K : B \subset A\}$ . By theorem 2.9 we denote  $T_2 = \{B \in K^{-1} : \exists C \in (K^{-1})^{-1} : C \subset B\}$ . It can be seen that  $T_1 \neq \emptyset$  and  $T_2 \neq \emptyset$ . If  $T_1 \cap T_2 = \emptyset$  then by the definition of  $K^{-1}$  there is  $C_1 \in (K^{-1})^{-1}$  such that  $T_2 \cup \{C_1\}$  is Sperner-system.

This contradicts to  $K^{-1}$  is saturated. Hence  $T_1 \cap T_2 \neq \emptyset$ . That is  $K \cup (K^{-1})^{-1}$  is not proper saturated.

The second case:  $K^{-1}$  isn't saturated. We can find a set  $A$  ( $A \in K$ ) and a set  $C \in (K^{-1})^{-1}$  such that  $A \subset C$  (Similar to the theorem 2.11). Consequently  $K \cup (K^{-1})^{-1}$  isn't proper saturated. The proposition is proved.

We can find the example which show that  $K \cup (K^{-1})^{-1}$  is saturated.

Remark 2.14. Let  $K$  be an arbitrary Sperner-system.

Basing on the algorithm for finding the set of antikeys we decide whether  $K$  is or isn't united (inclusive, saturated).

### §3. DEPENDENCIES

The functional, dual dependencies were shown in [4,6].

Definition 3.1. Let  $\Omega = \{1, \dots, n\}$  and  $P(\Omega)$  its power set. Let  $R$  be a relation over  $\Omega$  and  $A, B \in P(\Omega)$ . We say that  $B$  functional depends on  $A$  in  $R$  if

$$(\forall g, h \in R)((\forall a \in A)(g(a)=h(a)) \rightarrow (\forall b \in B)(g(b)=h(b)))$$

Denote  $A \stackrel{f}{\underset{R}{\rhd}} B$  and  $B$  dual depends on  $A$  in  $R$  if.

$$(\forall g, h \in R)((\forall a \in A)(g(a)=h(a)) \rightarrow (\forall b \in B)(g(b)=h(b)))$$

Denote  $A \stackrel{d}{\underset{R}{\rhd}} B$  and  $Y_R = \{(A, B) : A \stackrel{y}{\underset{R}{\rhd}} B\}$ , where  $Y \in \{F, D\}$ ,  $y \in \{f, d\}$ .

Definition 3.2. Let  $Y \subseteq P(\Omega) \times P(\Omega)$ . We say that  $Y$  satisfies the  $F$ -axiom if for all  $A, B, C, D \in P$

$$(F1) \quad (A, A) \in Y$$

$$(F2) \quad (A, B) \in Y, (B, C) \in Y \rightarrow (A, C) \in Y$$

$$(F3) \quad (A, B) \in Y, C \subseteq A, D \subseteq B \rightarrow (C, D) \in Y$$

$$(F4) \quad (A, B) \in Y, (C, D) \in Y \rightarrow (A \cup C, B \cup D) \in Y$$

$Y$  satisfies the  $D$ -axiom if for all  $A, B, C, D \in P(\Omega)$

$$(D1) \quad (A, A) \in Y$$

$$(D2) \quad (A, B) \in Y, (B, C) \in Y \rightarrow (A, C) \in Y$$

$$(D3) \quad (A, B) \in Y, C \subseteq A, B \subseteq D \rightarrow (C, D) \in Y$$

$$(D4) \quad (A, B) \in Y, (C, D) \in Y \rightarrow (A \cap C, B \cap D) \in Y$$

$$(D5) \quad (A, \emptyset) \in Y \rightarrow A = \emptyset.$$

In [4,6] the following theorem was proved.

Theorem 3.3. [4,6]. Let  $Y \subseteq P(\Omega) \times P(\Omega)$  and  $Y \in \{F, D\}$ ,  $Y' \in \{F, D\}$ .  $Y$  satisfies the  $Y'$ -axiom if and only if there exists a relation  $R$  over  $\Omega$  such that  $Y = Y_R$ .

Definition 3.4. Let  $Y \subseteq P(\Omega) \times P(\Omega)$ .

We say that  $Y$  satisfies the  $A$ -axiom if for every  $A \in P(\Omega)$ . There exists  $E(A)$  such that (1)  $A \subseteq E(A)$  and  $\forall B \subseteq E(A) \rightarrow (A, B) \in Y$

$$(2) (C, D) \in Y, C \subseteq E(A) \rightarrow D \subseteq E(A)$$

$Y$  satisfies the  $B$ -axiom if for every  $B \in P(\Omega)$  there is  $E(B)$  such that (1)  $B \subseteq E(B)$  and  $\forall A \subseteq E(B) \rightarrow (A, B) \in Y$

$$(2) (C, D) \in Y \text{ and } C \subseteq E(B) \rightarrow D \subseteq E(B)$$

Theorem 3.5. Let  $F(D) \subseteq P(\Omega) \times P(\Omega)$

$F(D)$  satisfies the  $A$ -( $B$ -)axiom if and only if there is a relation  $R$  over  $\Omega$  such that  $F = F_R$  ( $D = D_R$ ).

Proof. By the theorem 3.3. we only must prove that  $F(D)$  satisfies the  $A$ -( $B$ -) axiom if and only if  $F(D)$  satisfies the  $F$ -( $D$ -) axiom.

Now suppose that  $F$  satisfies the  $A$ -axiom. Then

(F1) It is obvious that  $(A, A) \in F$

(F2) If  $(A, B) \in F$ ,  $(B, C) \in F$  then by (1) and (2) we have  $C \subseteq E(A)$  and  $(A, C) \in F$ .

(F3) If  $(A, B) \in F$  and  $A \subseteq A'$ ,  $B' \subseteq B$  then  $A \subseteq A'$   $\subseteq E(A')$  implies  $B' \subseteq B \subseteq E(A')$  and by (1) we have  $(A', B') \in F$ .

(F4) If  $(A, B) \in F$  and  $(C, D) \in F$  then by (1) there is  $E(A \cup C)$  such that  $(A \cup C) \subseteq E(A \cup C)$  and  $\forall X \subseteq E(A \cup C) : (A \cup C, X) \in F$ .  $A \cup C \subseteq E(A \cup C)$  implies  $A \subseteq E(A \cup C)$ . By  $(A, B) \in F$  we have  $B \subseteq E(A \cup C)$ .  $C \subseteq E(A \cup C)$  and  $(C, D) \in F$  imply  $D \subseteq E(A \cup C)$ . So  $B \cup D \subseteq E(A \cup C)$  and by (1) we have  $(A \cup C, B \cup D) \in F$ .

Conversely,  $F$  satisfies the  $F$ -axiom and  $A \in P(\Omega)$ . We define  $E(A) = \{a \in \Omega : (A, a) \in F\}$ . It is obvious that  $E(A)$  satisfies (1). If  $(C, D) \in F$  and  $E(A) \supseteq C$ , but  $D \not\subseteq E(A)$  then  $(E(A), E(A) \cup D) \in F$ . Hence  $(A, E(A) \cup D) \in F$ . This contradicts to the definition of  $E(A)$ . Consequently  $E(A)$  satisfies (2). For the  $B$ -axiom we denote  $F' = \{(B, A) : (A, B) \in D\}$ . By the duality of  $F$ -axiom and  $D$ -axiom we only must prove that  $F$  satisfies the  $A$ -axiom  $\leftrightarrow D$  satisfies the  $B$ -axiom.

For every  $B \in P(\Omega)$  we construct  $E(B)$  similar to  $E(A)$ .

It is clear that  $E(B)$  satisfies (1) and (2) in the  $B$ -axiom. Conversely, it can be seen that  $D$  satisfies the  $B$ -axiom then  $F$  satisfies the  $A$ -axiom. The theorem is proved.

Definition 3.6. Let  $F \subseteq P(\Omega) \times P(\Omega)$  and  $F$  satisfies  $F$ -axiom. We say that  $(A, B) \in F$  is a maximal element of  $F$  if for all  $A', B'$  ( $A' \subseteq A, B' \subseteq B$ ) and  $(A', B') \in F$  then  $A' = A$  and  $B' = B$ .

Denote  $S(F)$  is set of maximal element of  $F$ .

We denote  $T(F) = \left\{ \begin{matrix} (A, B) \in F \\ A \subseteq B \end{matrix} : \forall (C, D) \in F : (C \subseteq B \rightarrow D \subseteq B) \text{ and } (A \subseteq D \rightarrow C \subseteq A) \right\}$

Theorem 3.7.  $S(F) = T(F)$ .

Proof. Suppose that  $(A, B) \in S(F)$  but  $(A, B) \notin T(F)$ . So  $\exists (C, D) \in F$  whether  $C \subseteq B$  but  $D \not\subseteq B$  (1) or  $A \subseteq D$  but  $C \not\subseteq A$  (2).

for(1):  $D \not\subseteq B$  implies  $D \neq \emptyset$  and  $D \cup B \supset B$ .  $C \subseteq B$  and  $(A, B) \in F$  implies  $(A, C) \in F$ .  $(C, D) \in F$  implies  $(A, D) \in F$ . Hence  $(A, B \cup D) \in F$ . This contradicts to  $(A, B) \in S(F)$ .

for(2):  $(C, D) \in F$  and  $A \subseteq D$  implies  $(C, A) \in F$ . Hence  $(C, B) \in F$ . Consequently, there is  $C \not\subseteq A$ :  $(C, B) \in F$ . This contradicts to  $(A, B) \in S(F)$ .

Conversely, if  $(A, B) \in T(F)$  but  $(A, B) \notin S(F)$ . That is whether  $\exists B' (B \subset B') : (A, B') \in F$  (1) or  $\exists A' (A \subset A') : (A', B) \in F$  (2).

It can be seen that (1) and (2) contradict to  $(A, B) \in T(F)$ . The theorem is proved.

#### ACKNOWLEDGEMENT

The author would like to take this opportunity to express deep gratitude to Professor DR Demetrovics János for his help, valuable comments and suggestions.

REFERENCES

- [1] W.W.Armstrong; Dependency Structures of Data base Relationships. Information Processing 74, North-Holland Publ. Co. (1974) 580-583.
- [2] G.Burosch, J.Demetrovics, G.O.H.Katona; The poset of closures. Order (1986)
- [3] E.F.Codd; Relational model of data for large shared data banks. Communications of the ACM, 12 (1970) 377-384.
- [4] G.Czédli, Függőségek relációs adatbázis modellben. Alk. Mat. Lapok (1980).
- [5] J.Demetrovics; On the equivalence of condidate keys with Sperner Systems. Acta Cybernetica, 4 (1979) 247-252.
- [6] J.Demetrovics; Reláció adatmodell logikai és strukturális vizsgálata. MTA-SZTAKI Tanulmányok, Budapest 114 (1980). 1-97.
- [7] J.Demetrovics, Z.Füredi, G.Katona; A függőségek és az individumok száma közötti kapcsolat összetett adatrendszerek esetén, Alkalmazott Matematikai Lapok, 9 (1983) 13-21.
- [8] Vu Duc Thi; Remarks on closure operations. MTA-SZTAKI Közlemények, Budapest, 30 (1984) 73-87.

Néhány speciális Sperner rendszer

VU DUC THI


Összefoglaló

Az adatbázisok leginkább sokoldalú modellje a Codd [3] által bevezetett relációs modell. Ezen modell vizsgálatában fontos szerepet játszanak a minimális kulcsok. Az [5]-ben a szerző bebizonyította, hogy a minimális kulcsok ekvivalensek a Sperner rendszerekkel. Ebben a cikkben a szerző néhány speciális Sperner rendszert és velük kapcsolatos funkcionális függőségeket vizsgál.

Специальные спернер системы

Бу Дук Тхи

Резюме



Реляционная модель данных, введенная Коддом [3], является одним из самых многообразных средств обработки данных. Она выдвигает на первый план не машинную эффективность, а наглядное описание данных с точки зрения пользователя.

В [5] доказано, что минимальные ключи эквивалентны Спернер системам.

В настоящей работе изучаются некоторые специальные Спернер системы и функциональные зависимости.





