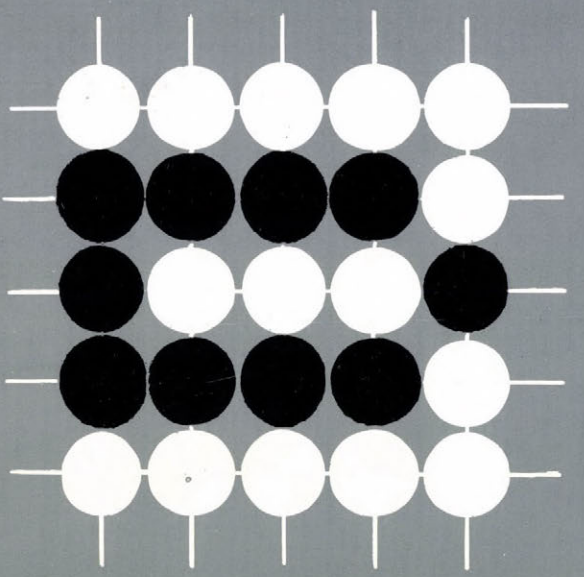
közlemények 33/1985

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest



Felelős kiadó:

Dr. VÁMOS TIBOR

ISBN 963 311 202 8
ISSN 0133-7459

Magyar Tudományos Akadémia
Számitástechnikai és Automatizálási Kutató Intézete

K Ö Z L E M É N Y E K

33/1985

C O N T E N T S

T A R T A L O M

С О Д Е Р Ж А Н И Е

MTA SZTAKI Közlemények 33/1985 p. 7-27

ТАБЛИЦЫ КОНЪЮНКТИВНЫХ ЗАПРОСОВ И ИХ ПРИМЕНЕНИЕ

Галя Ангелова

Лаборатория математической лингвистики
Институт математики с ВЦ, БАН
София, Болгария

1. ВВЕДЕНИЕ

Понятие "таблица" введено в /1/ и /7/ как средство изучения класса реляционных запросов, которые в реляционной алгебре представлены при помощи выражений, содержащих только операции σ (селекцию), π (проекцию) и J (соединение). Это так называемые SPJ - выражениям, составляющие важный класс выражений в реляционной алгебре. Каждому SPJ - выражению поставлена в соответствие таблица, а путем преобразования этой таблицы возможно получить множество SPJ - выражений, эквивалентных данному; возможно также выделить среди них то SPJ - выражение, которое содержит наименьшее число J - операций и таким образом оптимизировать начальное SPJ - выражение по отношению числа J - операций. При помощи понятия таблицы можно представить необходимое и достаточное условие сильной и слабой эквивалентности двух заданных SPJ - выражений. Так как между конъюнктивными реляционными запросами и SPJ - выражениями существует взаимно-однозначное соответствие, то при помощи таблиц конъюнктивных запросов можно исследовать проблему сильной и слабой эквивалентности данных конъюнктивных запросов. Понятие таблыци играет важную роль и в алгоритме интерпретации запросов пользователя в System/U (/6/ и /7/), где оно применяется для нахождения оптимального внутреннего представления запроса, которое является слабоэквивалентным начальному представлению запроса.

2. ДЕФИНИЦИИ И ПРИМЕРЫ

Будем использовать дефиниции операций селекции, проекции и (естественного) соединения:

Пусть r - отношение над множеством атрибутов X и $A \in X$, $a \in A$, $Y \subseteq X$. Тогда:

- селекция $A \theta a$, обозначаемая через $\sigma_{A \theta a}(r)$, представляет собой:

$$\sigma_{A \theta a}(r) = \{ \alpha / \alpha \in r \text{ и } \alpha(A) \, \theta \, a \}.$$

(Здесь θ - одно из $=, <, >, \leq, \geq, \neq$). Таким образом из отношения r берутся только те кортежи, имеющие в атрибуте A значение b и $b \theta a$. Так $\sigma_{A \theta a}(r)$ является отношением над множеством атрибутов X и следовательно представляет собой подмножество отношения r;

- проекция $\Pi_Y(r)$ представляет собой:

$$\Pi_Y(r) = \{ \beta[Y] / \beta \in r \},$$

т.е. из всех возможных кортежей отношения r берутся только значения атрибутов множества Y и одинаковые кортежи отождествляются. Так $\Pi_Y(r)$ является отношением над множеством атрибутов Y;

- (естественное)соединение $r_1 \bowtie r_2$.

Пусть R_1 и R_2 являются реляционными схемами, а r_1 и r_2 - отношения над этими реляционными схемами. Тогда

$$r_1 \bowtie r_2 = \{ \gamma / \gamma \text{ является кортежом над атрибутами } R_1 \cup R_2 \text{ и существуют кортежи } v_1 \in r_1 \text{ и } v_2 \in r_2, \text{ такие, что } v_1 = \gamma[R_1] \text{ и } v_2 = \gamma[R_2] \}.$$

Дефиниция 1. SPJ - выражения будем называть выражениями реляционной алгебры, если:

а) операнды выражений являются реляционными схемами;

б) операции выражений представляют собой селекцию, проекцию и (естественное) соединение,

т.е. эти выражения являются формулами над S , P , J и именами реляционных схем.

Дефиниция 2. Конъюнктивным запросом в реляционном языке запросов будем называть запрос вида:

(1) $\{a_1 a_2 \ldots a_n / (\exists b_1)\ldots(\exists b_m)\ (P_1 \wedge P_2 \wedge \ldots \wedge P_k)$

где P_i, $1 \leq i \leq k$ – термы вида а) или вида б):

а) $R(c_1 c_2 \ldots c_s)$, что означает, что кортеж $c_1 c_2 \ldots c_s$ принадлежит отношению над реляционной схемой R . Здесь c_j , $1 \leq j \leq s$ являются константами соответствующего домена или $c_j \in \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m\}$ – (т.е. c_j – символы среди $a_1, a_2, \ldots, a_n, b_1, \ldots, b_m$);

б) $c \theta d$, где c и d – либо константы, либо элементы множества $\{a_1, a_2, \ldots, a_n, b_1, \ldots, b_m\}$. Здесь θ – одно из $=$, $<$, $>$, $\leq$, $\geq$.

Пример 1. Рассмотрим следующую базу данных, состоящую из пяти примерных реляционных схем:

ЧАСТЬ (ЧИМЯ, ЧНОМЕР, ЦЕНА)
ПОСТАВЩИК (ПИМЯ, ПНОМЕР, ПАДРЕС, ПГОРОД)
КЛИЕНТ (КИМЯ, КНОМЕР, КАДРЕС, КГОРОД)
ПОСТАВКА (ЧНОМЕР, ПНОМЕР, КНОМЕР, КОЛИЧЕСТВО)
ОБЯЗАННОСТЬ(ЧНОМЕР, ПНОМЕР).

Реляционные схемы нормализованы в третьей нормальной форме. Отношение ОБЯЗАННОСТЬ дает информацию об обязанностях, присущих каждому поставщику.

В качестве примера конъюнктивного запроса к этой базе данных можно рассмотреть запрос:

q_1 : Найти имена всех поставщиков, живущих в городе c_1 .

Этот запрос можно представить и следующим образом:

(2) $\{a_1 / (\exists b_1)(\exists b_2) : \text{ПОСТАВЩИК}(a_1, b_1, b_2, c_1)\}$.

Как известно, между конъюнктивными запросами и SPJ - выражениями существует взаимно-однозначное соответствие, т.е. каждый конъюнктивный запрос может быть представлен как SPJ - выражение и наоборот, каждому SPJ - выражению соответствует конъюнктивный запрос (см. /7/). По этой причине мы будем строить таблицы для SPJ - выражений и часто будем интерпретироват эти таблицы как конъюнктивные запросы.

Дефиниция 3. Введем понятие таблицы:
Каждая таблица представляет собой двумерную матрицу, причем к этой матрице можно задавать и список ограничений. Столбцы матрицы соответствуют заданному множеству атрибутов - A_1, A_2 ,..., A_k , порядок которых фиксирован. Таблица может содержить произвольное число строк; ее элементы - символы следующих видов:

а) свободные переменные (distinguished variables) - они соответствуют $a_1, a_2, \ldots, a_n$ в (1) и (2). Будем обозначать их через букву a с нижным индексом - a_1 , a_2 ,....;

б) Связанные переменные (nondistinguished variables)- они соответствуют символам $b_1, b_2, \ldots, b_m$ в (1) и (2). Будем обозначать их через букву b с нижным индексом - b_1 , b_2 ,....;

в) константы - полагается, что константы, находящиеся в j -том столбце, принадлежат домену, соответствующему атрибуту A_j ;

г) пробелы.

Над таблицей (или в качестве ее первой строки) задаются атрибуты, для которых составлена данная таблица - A_1, A_2 ,..., A_k . В следующей строке (здесь мы будем считать, что именно она является первой строкой таблицы) могут находиться только свободные переменные, константы или пробелы.

Эта строка называется резюме таблицы и представляет собой выражение, находящееся в (1) и (2) слева от косой черты "/". Способ расположения свободных переменных в резюме таблицы показывает к каким атрибутам следует их отнести. Например, a_1a_2 не означает, что a_1 является свободной переменной над атрибутом A_1, а a_2 - свободной переменной над атрибутом A_2. Запись a_1a_2 обретает смысл только в конкретной таблице, причем расположение переменных a_1 и a_2 в резюме показывает к каким атрибутам относятся эти две переменные. Остальные позиции резюме - пустые или содержат константы.

Кроме строки резюме таблица содержит и строки, описывающие выражения справа от косой чертой в (1) и (2). Эти строки будем называть просто "строками" таблицы и будем их использовать для описания термов вида $R(c_1,c_2,\ldots,c_s)$ в п. а) дефиниции 2. Каждому терму $R(c_1,\ldots,c_s)$ отводим одну строку таблицы следующим способом:

- если отношение R задано над атрибутами $A_{i_1},A_{i_2},\ldots,A_{i_s}$, то тогда в столбцы, соответствующие этим атрибутом, ставим c_1 для A_{i_1}, c_2 для $A_{i_2},\ldots,c_s$ для A_{i_s}. Из п.а) дефиниции 2 видно, что таким образом в строке могут участвовать свободные переменные, связанные переменные и константы;
- в столбцы атрибутов, которые не участвуют в отношении R, ставим пробелы.

Каждой строке ставим маркер с правой стороны таблицы - если строка отведена терму $R(c_1,c_2,\ldots,c_s)$, справа ставим маркер (R) и таким образом отмечаем "откуда" берется эта строка.

Видно, что при этом построении резюме и строк таблицы переменные участвуют только в столбцах атрибутов, к которым они относятся - т.е. одна переменная не может фигурировать

одновременно в двух разных столбцах. Кроме того требуется, чтобы свободная переменная не появлялась в строках таблицы, если она не фигурирует в ее резюме.

Таким образом при помощи таблицы мы описали выражения слева от косой черты в (1) и (2) и термы вида $R(c_1, c_2, \ldots, c_s)$, находящиеся справа от косой черты. Так как справа в (1) могут фигурировать и выражения вида $c \vartheta d$ (п.б/ дефиниции 2), каждый терм вида $c \vartheta d$ записывается под строками таблицы. Таким образом формируется список ограничений, который тоже рассматривается как часть таблицы.

Пример 2. Для выражения (2) над базой данных из примера 1 получаем таблицу

(3')

ПИМЯ	ПНОМЕР	ПАДРЕС	ПГОРОД	
a_1				
a_1	b_1	b_2	b_3	(ПОСТАВЩИК)

$b_3 = c_1$

или

(3'')

ПИМЯ	ПНОМЕР	ПАДРЕС	ПГОРОД	
a_1				
a_1	b_1	b_2	c_1	(ПОСТАВЩИК)

Результатом таблицы (а также результатом конъюнктивного запроса) является отношение. Это отношение-результат над атрибутами, содержащими свободные переменные в резюме данной таблицы.

Пример 3. Для таблицы 3'' отношением-результатом является:

$R' = \{a_1 / a_1 \in$ ПИМЯ и существуют значения b_1 атрибута ПНОМЕР и b_2 атрибута ПАДРЕС такими, что

кортеж $a_1 b_1 b_2 c_1$ принадлежит отношению ПОСТАВЩИК$\}$.

Здесь мы будем интерпретировать отношение R' как "результатом" таблицы 3''.

Рассмотрим другую примерную таблицу T_1:

T_1:

A_1	A_2	A_3	A_4	A_5	
	a_1		a_2		
b_1	a_1	b_2			(R_1)
		b_2	a_2	b_3	(R_2)
	b_4		b_5	c_1	(R_3)

$b_3 < c_1$

Отношение-результат можно записать следующим образом

$$R(T_1) = \{a_1 a_2 \ / \ a_1 \in A_2, a_2 \in A_4 \text{ и существуют } b_1 \in A_1, b_2 \in A_3, b_3 \in A_5, b_4 \in A_2, b_5 \in A_4 \text{ такие, что } b_1 a_1 b_2 \in R_1 \text{ и } b_2 a_2 b_3 \in R_2 \text{ и } b_4 b_5 c_1 \in R_3 \text{ и } b_3 < c_1\}.$$

Легко представить запись конъюнктивного запроса, для которого составлена таблица T_1:

$$\{a_1 a_2 \ / \ (\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4)(\exists b_5) \text{такие, что } b_3 < c_1 \wedge R_1(b_1\ a_1\ b_2) \wedge R_2(b_2\ a_2\ b_3) \wedge R_3(b_4\ b_5\ c_1)\}.$$

Задавая более сложные запросы, мы часто представляем в виде столбца таблицы все атрибуты, участвующие в реляционных схемах конкретной базы данных. Так как таблица используется и для описания конъюнктивных запросов в универсальных реляционных системах, для таких таблиц приходится перечис-

лять столбцы всех атрибутов универсального отношения. В связи с этим нужно отметить некоторые особенности процесса стождествления разных атрибутов как один столбец данной таблицы.

Рассмотрим следующий запрос к примерной базе данных:

q_2 : Найти имена всех поставщиков и всех клиентов, живущих в одном и том же городе.

Конъюнктивное представление запроса:

(4) $\{a_1 a_2 / (\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4)(\exists b_5)(\exists b_6)$ так, что

ПОСТАВЩИК $(a_1 b_1 b_2 b_3)$ и КЛИЕНТ $(a_2 b_4 b_5 b_6)$

и $(b_3 = b_6)\}$.

Соответствующая таблица имеет вид:

T_2:

ПИМЯ	ПНОМЕР	ПАДРЕС	ПГОРОД	КИМЯ	КНОМЕР	КАДРЕС	КГОРОД
a_1				a_2			
a_1	b_1	b_2	b_3				(ПОСТАВЩИК)
				a_2	b_4	b_5	b_6 (КЛИЕНТ)

$b_3 = b_6$

В этой примерной базе данных атрибуты ПАДРЕС и ПГОРОД изменяются в тех же доменах, в которых соответственно изменяются КАДРЕС и КГОРОД. Представляется очень заманчивым объединить их в виде двух столбцов таблицы с именами напр. АДРЕС и ГОРОД, как это сделано в примере 8.6 в /7/. Тогда для q_2 мы бы получили таблицу:

T_2' :

ШИМЯ	ШНОМЕР	АДРЕС	ГОРОД	КИМЯ	КНОМЕР	
a_1				a_2		
a_1	b_1	b_2	b_3			(ПОСТАВЩИК)
		b_4	b_3	a_2	b_5	(КЛИЕНТ)

В случае допущения такого отождествления атрибутов в столбцах таблицы могут возникнуть проблемы в процессе построения таблицы для запроса q_3 :

q_3 : Найти адреса всех поставщиков и всех клиентов, живущих в одном и том же городе.

Его конъюнктивная запись имеет вид:

(5) $\{a_1a_2/(\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4)(\exists b_5)$ так, что

ПОСТАВЩИК $(b_1b_2a_1b_3) \wedge$ КЛИЕНТ $(b_4b_5a_2b_6) \wedge$
$\wedge (b_3 = b_6)\}$.

В этом случае невозможно записать (5) в виде таблицы со столбцами, как таблицу T_2', так как мы нуждаемся в двух свободных переменных, которые нужно внести в столбец АДРЕС (что согласно дефиниции понятия таблицы не является возможным). Нам необходима таблица, столбцы которой должны выглядеть как столбцы таблицы T_2.

Следовательно можно заключить, что при отождествлении атрибутов и столбцов таблиц нужно соблюдать т.наз. предположение о единственной роли (unique role assumption - см. /3/). В этом случае можем быть уверены, что данная выше дефиниция таблицы позволить нам сопоставлять каждому конъюнктивному запросу соответствующую ему таблицу.

3. ПОСТРОЕНИЕ ТАБЛИЦ ПО ДАННЫМ SPJ - ВЫРАЖЕНИЯМ

Дефиниция 3 показывает построение таблицы по данному конъюнктивному запросу. Рассмотрим алгоритм построения таблицы для данного SPJ - выражения.

Значение каждого SPJ - выражения является отношением и его можно рассматривать в качестве ответа некоторого конъюнктивного запроса. Следовательно для данного SPJ - выражения мы можем построить таблицу, представляющую собой запрос, ответ которого - данное SPJ - выражение. Так как SPJ - выражение является формулой и его можно строить индуктивным образом, то таблицу SPJ - выражения также можно строить индуктивным образом. Достаточно показать способ сопоставления таблиц выражениям r, $\sigma_{A_i=c}(E_1)$, $\Pi_X(E_1)$ и $E_1 \bowtie E_2$, где E_1 и E_2 - SPJ - выражения, для которых мы получили таблицу.

Предположим, что $E = r$. Тогда таблица состоит из резюме и еще одной строки. В столбцах, соответствующих именам атрибутов схемы r, в резюме находятся свободные переменные. Другие столбцы в резюме пустые. В строке в столбцах, соответствующих именам атрибутов схемы r, находятся те же самые свободные переменные, которые находятся и в резюме; другие столбцы этой строки заполнены разными связанными переменными. Таблица для отношения ЧАСТЬ примерной базы данных дана на фиг. 2.

Предположим, что $E = \sigma_{A_i=c}(E_1)$. Тогда, если T_1 - таблица для E_1, таблицу T для E можно получить из T_1 следующим способом:

а) если столбец A_i в резюме T_1 - пустой, то выражение E не имеет смысла и таблица T - неопределена.

б) если в столбце A_i в резюме T_1 имеется константа c_1, то таблица T совпадает с T_1,

если $c_1 = c$; в противном случае $T = \emptyset$.

в) если в столбце A_i в резюме T_1 имеется свободная переменная a, то таблица T получается от таблицы T_1 путем замещения a через c, независимо от того, в каком месте встечается a в T_1.

Получение таблицы для операции селекцию илюстрировано на фиг. 2.

Предположим, что $E = \Pi_X(E_1)$, причем T_1 - таблица для E_1. Таблицу T для E строим из таблицы T_1 для E_1 следующим образом: в резюме T_1 ставим "пустые символи" в столбцы, непринадлежащие X. Во всех остальных строках для этих столбцов свободные переменные заменяются разными связанными переменными.

Предположим, что $E = E_1 \bowtie E_2$ и T_1 и T_2 - таблицы для E_1 и E_2 соответственно. Без потери общности можно предположить, что множества связанных переменных T_1 и T_2 не пересекаются и что если в одних и тех же столбцах в строках резюме T_1 и T_2 фигурируют свободные переменные, то они являются одинаковыми. Таблицу T для E конструируем следующим способом: если в резюме T_1 и T_2 на одном и том же месте фигурируют разные константы, то $T = \emptyset$. В противном случае строками T являются строки T_1 и T_2; причем резюме T образовано из резюме T_1 и T_2 как следует. Если в данном столбце A_i фигурируют:

а) константа c в резюме одной из таблиц T_1 и T_2, то в резюме T ставится c и везде свободная переменная другой таблицы заменяется константой c;

б) свободная переменная в одной из таблиц или в обеих, то в резюме T ставится та же переменная;

в) пустые символы в обеих таблицах T_1 и T_2,

то в таблицу T ставим также пустой символ.

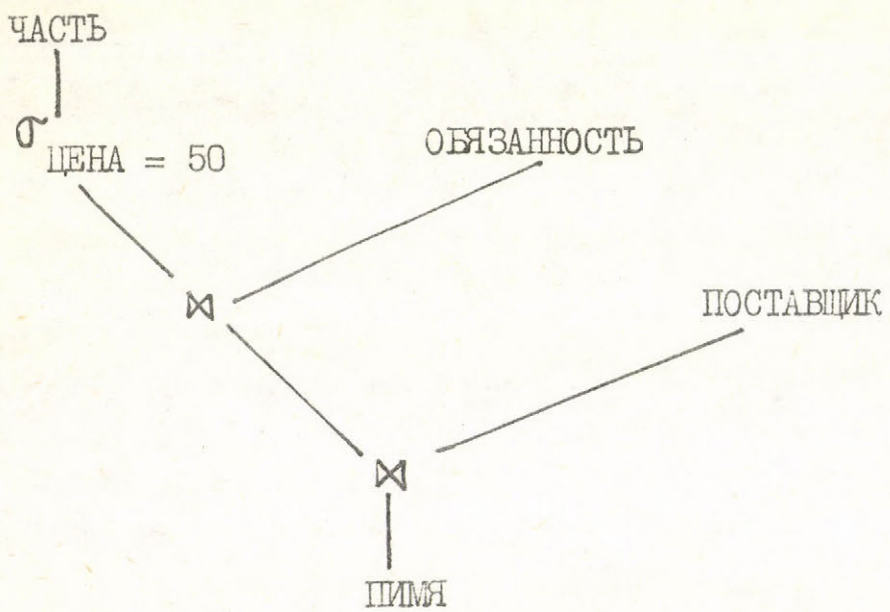
Пример 4. Проиллюстрируем процесс конструирования таблицы для SPJ - выражения на следующем примере:

q_4 : Найти имена поставщиков, поставляющие части ценой 50.

Ответ запроса можно представить при помощи SPJ - выражения:

$$\Pi_{\text{ПИМЯ}}\,((\text{ОБЯЗАННОСТЬ} \bowtie \sigma_{\text{ЦЕНА}\,=\,50}\,(\text{ЧАСТЬ})) \bowtie \text{ПОСТАВЩИК}). \qquad (6)$$

Дерево разбора /2/ этого выражения дано на фиг. 1.



Фиг. 1. Дерево разбора для SPJ- выражения (6).

При помощи этого дерева представляется последовательность конструирования таблицы для SPJ - выражения (6). В таблицах на фиг. 2 атрибуты обозначены только через две буквы (например, ЧИМЯ обозначено через ЧИ - $\frac{Ч}{И}$), а все связанные переменные, встечающиеся только один раз, пропущены (т.е. замещены пустыми символами).

4. ЭКВИВАЛЕНТНОСТЬ И МИНИМИЗАЦИЯ ТАБЛИЦ

Как указано выше, понятие таблицы вводится с целью исследовать эквивалентность конъюнктивных запросов, полагая, что таким образом запрос легче поддается формализации. Как следует ожидать, два запроса являются эквивалентными тогда и только тогда, когда их таблицы эквивалентны.

Использование понятия таблицы основывается на следующих дефинициях:

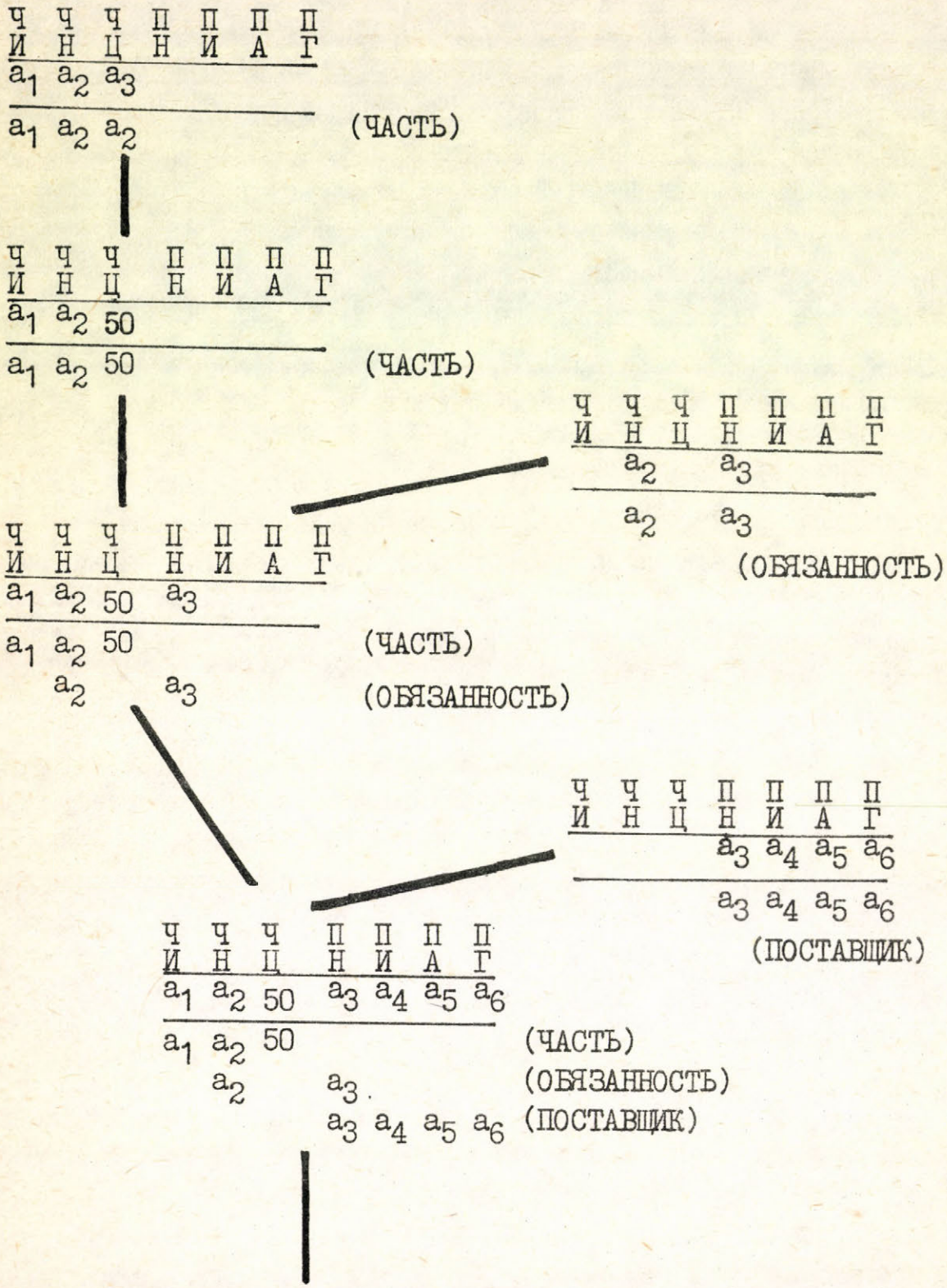
Дефиниция 4. Пусть $d = \{r_1, r_2, .., r_n\}$ представляет собой состояние базы данных, т.е. множество отношений над реляционными схемами $\{R_1, R_2, .., R_n\}$. Тогда отношение - результат, сопоставленное данной таблице T при помощи вышеописанной интерпретации, будем означать через T(d) (естественно, значение этого отношения является различным для различных состояний d).

Дефиниция 5. Будем говорить, что $T_1 \subseteq T_2$, если для каждого d в силе $T_1(d) \subseteq T_2(d)$.

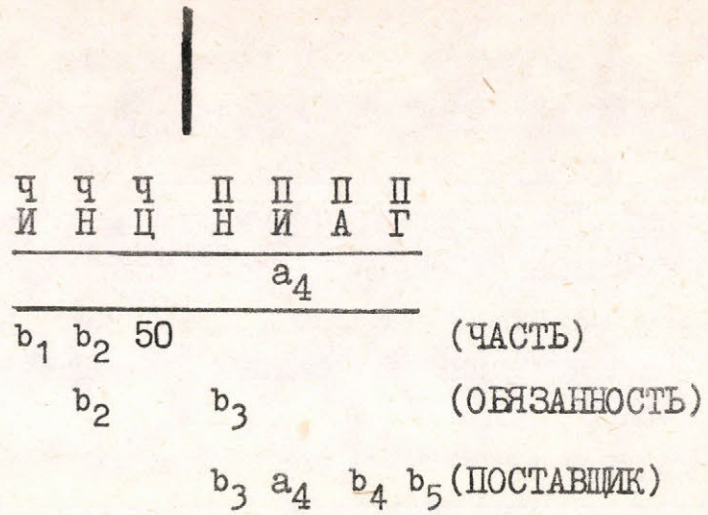
Дефиниция 6. T_1 и T_2 являются эквивалентными ($T_1 \equiv T_2$) тогда и только тогда, когда $T_1 \subseteq T_2$ и $T_2 \supseteq T_1$

В основе алгоритма для определения эквивалинтности двух данных таблиц (см. /1/ и /7/), лежит понятие "отображения" (mapping) между символами и строками таблиц.

Дефиниция 7. Пусть T_1 и T_2 - таблицы. Отображение h символов T_1 на символы T_2 называется "содержащим стображением" (containment mapping), если:



Фиг. 2. Конструирования таблицу для SPJ - выражения (6).



Фиг. 2. (Продолжение). Конструирование таблицу для SPJ- выражения (6).

а) h отображает символы резюме T_1 в символы резюме T_2 ;

б) h отображает символы любой строки T_1 в символы строки T_2 с таким же маркером, как у строки из T_1 . При этом h сохраняет значение всех констант.

в) h отображает список ограничений T_1 в множество ограничений, являющееся подмножеством ограничений T_2 .

Если h отображает все символы строки в символы другой строки, говорим, что h отображает всю данную строку в другую.

Таким образом будем рассматривать h и как отображение одного символа в другой, и как отображение одной строки в другую.

Теорема 1. (см. /7/). $T_1 \supseteq T_2$ тогда и только тогда, когда существует "содержащее отображение" h из T_1 на T_2 .

Следствие. $T_1 \equiv T_2$ тогда и только тогда, когда существует "содержащее отображение" h_1 из T_1 на T_2 и h_2 из T_2 на T_1.

Дефиниция 8. Пусть T - таблица. Минимальной таблицей для таблицы T будем называть таблицу, содержащую минимальное число строк и эквивалентной таблице T. Процесс нахождения минимальной таблицы для данной таблицы T будем называть оптимизацией таблицы T.

В рассматриваемых до сих пор таблицах маркеры (tags) показывают из какого отношения берется данная строка. Определенная выше эквивалентность, где в п.б) дефиниции 7 требуется сохранить маркер строки при отображении одной таблицы в другую, называется сильной эквивалентностью. Если мы поставим себе задачу оптимизировать число J - операций в процессе реализации данного конъюнктивного запроса, то пользуясь техникой таблиц, можем найти таблицу, эквивалентную данной, содержащую минимальное число строк (эта задача является NP - complete). Эта минимальная таблица, сильно эквивалентная данной таблице, должна содержить строки с такими же маркерами, как и выходная таблица. Такая минимальная таблица предполагает, что при обработке начального конъюнктивного запроса будут применяться J - операции ко всем отношениям, упомянутым в первоначальной формулировке запроса. Однако это не всегда является необходимым. Требование рассматривать строки таблиц вместе с их маркерами не в силе, если предположить существование универсального отношения U над атрибутами

$$R_1 \cup R_2 \cup \ldots \cup R_n ,$$

такого, что $r_i = \pi_{R_i}(U)$ для $1 \leq i \leq n$. Это предположение известно под именем "предположение существования универсума" (universal instance assumption- см. /3/). В таком случаю каждая строка таблиц для конъюнктивных запросов снабжена маркером U, т.е. каждая строка берется из универсума.

Таким образом не нужно учитывать откуда взялась каждая строка и маркеры могут быть пропущены. Тогда при оптимизации таблицы возможно исчезновение всех строк с маркерами R_s для некоторого отношения r_s, которые присутствовали в первоначальном представлении таблицы. Предположение о существовании универсума ведет к определении понятия слабой эквивалентности.

Дефиниция 9. Пусть E_1 и E_2 – два запроса над данным состоянием d и пусть U – универсальное отношение для d. E_1 и E_2 являются слабо эквивалентными (записываем $E_1 \equiv_w E_2$), если $E_1(U) = E_2(U)$ для каждого возможного состояния U.

Аналогичную дефиницию вводим и для таблиц.

Дефиниция 10. Пусть T_1 и T_2 – таблицы соответственно для конъюнктивных запросов E_1 и E_2. T_1 и T_2 являются слоба эквивалентными (записываем $T_1 \equiv_w T_2$) тогда и только тогда, когда $E_1 \equiv_w E_2$.

В /7/ показано, что необходимое и достаточное условие слабой эквивалентности двух таблиц T_1 и T_2 – данное в теореме 1 условие. Как мы уже отметили, в этом случае отображения h_1 и h_2 не будут учитывать маркеры разных строк таблиц (так как U является маркером всех строк).

5. ИСПОЛЬЗОВАНИЕ ТАБЛИЦ КОНЪЮНКТИВНЫХ ЗАПРОСОВ

Так как техника таблицы дает нам возможность устанавливать эквивалентность двух конъюнктивных запросов, она может быть применена для нахождения оптимального представления запроса относительно данного критерия. Таким критерием является: вычислить значение данного SPJ- выражения, используя минимальное число J – операций (реализация J – операции достаточно тяжела).

Из алгоритма построения таблицы для данного SPJ - выражения видно, что операция J порождается парой строк в таблице. Следовательно для данной таблицы n строками можно конструировать реализацию соответствующему запросу при помощи n-1 J - операций. При таком критерие оптимальности проблема оптимизирования данного запроса сводиться к нахождению таблицы, слабо эквивалентной данной таблице.

Пример 5. Предположим, что для базы данных из примера 1 выполнено предположение о существовании универсума. Ищем ответ для следующего запроса:

q_5 : Найти имена всех поставщиков, поставляющие (или уже поставшие) части, количество которых - 500.

SPJ - выражение, реализирующее ответ:

(7) $\Pi_{\text{ПИМЯ}}$ (ПОСТАВЩИК ⋈ ОБЯЗАННОСТЬ ⋈ ПОСТАВКА).

Соответствующая этому запросу таблица T_5: (здесь пропущены столбцы атрибутов, не участвующих в описании q_5):

T_5 :

ПИМЯ	ПНОМЕР	ПАДРЕС	ПГОРОД	ЧНОМЕР	КНОМЕР	КОЛИЧЕСТ.
a_1						
a_1	b_1					
	b_1			b_2		
	b_1			b_2		500

Сразу видно, что T_6 является оптимальной слабо эквивалентной таблицей для таблицы T_5:

T_6:

ПИМЯ	ПНОМЕР	ПАДРЕС	ПГОРОД	ЧНОМЕР	КНОМЕР	КОЛИЧЕСТВО
a_1						
a_1	b_1					
	b_1					500

так как существуют отображение h_1 символов строк T_5 в символы строк T_6 и отображение h_2 символов строк T_6 в символы строк T_5. h_1 отображает первую строку T_5 в первую строку T_6, вторую строку T_5 в первую строку T_6 и последную строку T_5 во вторую строку T_6; h_2 отображает первую строку T_6 в первую строку T_5 и вторую строку T_6 в последную строку T_5.

Таблица T_6 представляет выражение

(8). $\pi_{\text{ПИМЯ}}$ (ПОСТАВЩИК $\bowtie$ ПОСТАВКА).

Таким образом ясно, что выражения (7) и (8) являются слабо эквивалентными.

Нужно отметить, что здесь допущение о существовании универсума является существенным. (Таблицы T_5 и T_6 не являются сильно эквивалентными).

Пример 5 иллюстрирует и применение понятия таблицы в System/U (см. /6/ и /7/). Так как в System/U основное предположение - предположение о существовании универсума, каждый конъюнктивный запрос пользователя представлен с помощью своей оптимальной таблицы (являющейся слабо эквивалентной данной таблице) и таким образом осуществляется более эффективная реализация запроса.

Другие применения техники таблиц даны например в /4/ и /5/.

6. ЗАКЛЮЧЕНИЕ

В настоящей работе описаны основные характеристики понятия таблицы и основные возможности его применения. Сейчас таблицы рассматриваются как средства для изучения конъюнктивных запроссв. Так как таблица дает синтезированное описание содержания некоторого отношения, ее можно применять и для изучения связей между разными типами зависимостей в рамках этого отношения.

ЛИТЕРАТУРА

1. Aho, A., Sagiv, Y., and Ullman, J. Efficient Optimization of a class of Relational Expressions. ACM TODS, Vol. 4, No. 4, December 1979, 435 - 454.
2. Maier D. The Theory of Relational Databases. Computer Science Press, 1983.
3. Maier, D., Ullman, J. and Vardi, M. On the Foundations of the Universal Relation Model. ACM TODS, Vol. 9, No. 2, June 1984, 283 - 308.
4. Mendelzon, A. Database States and Their Tableaux. ACM TODS, Vol. 9, No. 2, June 1984, 264 - 282.
5. Klug, A. and Price, R. Determining View Dependencies Using Tableaux. ACM TODS, Vol. 7, No. 3, September 1983, 361 - 380.
6. Korth, N., Kuper, G., Feigenbaum, J., Val Gelder, A. and Ullman, J. System/U: a Database System based on the Universal Relation Assumption. ACM TODS, Vol. 9, No. 3, September 1984, 331 - 347.
7. Ullman, J. Principles of Database Systems. Computer Science Press, 1982.

Konjuktiv lekérdezési táblázatok és alkalmazásaik

G. Angelova

Összefoglaló

A szerző bevezeti a "konjuktiv lekérdezési táblázat" fogalmát és megadja az ilyen táblázatokra vonatkozó alapvető definiciókat és jellemzéseket. Leir néhány alkalmazási lehetőséget is pl. az adott SPJ-kifejezésnek megfelelő táblázat konstruálására vonatkozó algoritmust vagy az adott táblázattal ekvivalens minimális táblázat megkeresését.
A táblázatok ekvivalenciájának feltételeivel is foglalkozik.

Tableaux and their applications

G. Angelova

Summary

The paper presents the notion of conjunctive query tableau. The basic definitions and characteristics of tableaux are given. The basic possibilities for applications are described: the algorithm for constructing tableau corresponding to a given SPJ - expression, the process of tableaux optimization in order to find a minimal tableau equivalent to a given one. The conditions for tableaux equivalence are described and thus an algorithm for finding weak or string equivalent conjunctive queries is presented.

MTA SZTAKI Közlemények 33/1985 p. 29-35

ИССЛЕДОВАНИЕ СТОХАСТИЧЕСКОЙ УСТОЙЧИВОСТИ АДАПТИВНЫХ НАБЛЮДАТЕЛЕЙ С ПОМОЩЬЮ ВЕКТОРНЫХ ФУНКЦИЙ ЛЯПУНОВА

Карабутов Н.Н.
Хадреви И.

МОСКОВСКИЙ ИНСТИТУТ СТАЛИ И СПЛАВОВ

Адаптивные наблюдатели /АН/ широко применяются в системах управления для решения задачи идентификации параметров и состояния объектов управления по экспериментальным данным о входе и выходе объекта. АН принято делить на два типа: явные и неявные в зависимости от того, каким образом в системе идентификации решается проблема оценки состояния. При синтезе адаптивных наблюдателей основное внимание уделяется обеспечению устойчивости системы идентификации. Получению условий, гарантирующих устойчивость АН, описываемых системой детерминированных дифференциальных уравнений, посвящено достаточно много работ. В отличие от этого исследованию качества работы адаптивных идентификаторов состояния в условиях действия случайных возмущений уделялось недостаточно много внимания. В основном изучалась проблема устойчивости АН явного типа [1—5], причем явного ограничения на интенсивность действующих помех, за исключением работы [5], получено не было.

В данной работе исследуется стохастическая устойчивость АН неявного типа линейных динамических объектов, уравнения которых приведены к неминимальной идентификационной форме, с помощью стохастического варианта метода векторных функций Ляпунова.

Рассмотрим вполне управляемый и наблюдаемый объект управления, описываемый уравнением

$$\tilde{x}^{(m)} + \tilde{a}_m \tilde{x}^{(m-1)} + \ldots + \tilde{a}_1 x = \tilde{b} u,$$

$$\tilde{y} = \tilde{x} + \xi^y, \qquad (1)$$

$$\tilde{u} = u + \xi^u,$$

где $\tilde{y}$, $\tilde{u}$ — измеряемый выход и вход объекта; $\tilde{a}_1 \ldots, \tilde{a}_m, \tilde{b}$ — неизвестные параметры; ξ^u, ξ^y — независимые центрированные случайные винеровские процессы типа белого шума

$$M\{\xi^u(t)\} = 0, \quad M\{\xi^u(t)\xi^u(\tau)\} = \sigma_u^2 \delta(t-\tau),$$

$$M\{\xi^y(t)\} = 0, \quad M\{\xi^y(t)\xi^y(\tau)\} = \sigma_y^2 \delta(t-\tau).$$

$M\{\cdot\}$ — знак математического ожидания, σ — интенсивность шумов измерения, δ — дельта-функция Дирака.

Считаем, что для входного воздействия $u(t)$ выполняется условие D_0: $u(t)$ содержит не менее $(m+1)/2$ синусоидальных составлющих, $u(t) \neq 0$, $|u(t)| \leqslant U < \infty$.

Для оценки неизвестных параметров объекта управления (1) применяется систе-

ма адаптивной идентификации, описываемая матричным стохастическим дифференциальным уравнением в форме Ито [6]:

$$dE = QE\,dt + D\sigma_y\,d\eta, \tag{2}$$

где $E(t) = [e_1(t) : \Delta A_3^{\tau}(t)]^{\tau}$; $E \in R^{2m}$; $e_1(t) = \hat{y}(t) - y(t)$; $\hat{y}(t)$ — выход АН, $\Delta A_3(t)$ — вектор рассогласования между параметрами АН и объекта, приведенного к неминимальной иденификационной форме [6, 7];

$\Delta A_3 = \hat{A}_3(t) - A_3$; $A_3 = [a_1, \ldots, a_m, b_2, \ldots, b_m]^{\tau}$ — вектор неизвестных параметров объекта в новом представлении.

Матрица $Q \in R^{2m \times 2m}$ и вектор $D \in R^{2m}$ задаются выражениями

$$Q = \left[\begin{array}{c|c} -\lambda & Z^{\tau} \\ \hline \Gamma Z & 0 \end{array}\right],$$

$$D^{\tau} = [\beta \,|\, (-\tilde{S}e_1\,dt + \tilde{Z}dt + \tilde{S}\sigma_y d\eta)^{\tau}\Gamma], \quad \tilde{S}^{\tau} = [1 \,|\, 0^{\tau}],$$

где $d\xi^y = \sigma_y d\eta$, η — независимый винеровский белый шум; $\lambda_1 > 0$ — некоторое число; $Z(t) \in R^{2m-1}$, $Z(t) = [y(t) : W_1^{\tau}(t) : W_2^{\tau}(t)]^{\tau}$; $W_1(t)$; $W_2(t)$ — векторы вспомогательных сигналов, получаемые путем пропускания выхода и входа объекта (1) через динамическую систему [6]

$$\dot{W}_i = \Lambda^{\tau} W_i + H_i\,\omega_i;\ i = 1, 2,\ \omega_1 = y(t);\ \omega_2 = u(t);$$

$$W_1(t_0) = W_{1\,0};\ W_2(t_0) = W_{20};\ \Lambda \in R^{(m-1)\times(m-1)} \quad —$$

— квазидиагональная матрица; $H_1 \in R^{m-1}$; $H_2 \in R^{m-1}$ — векторы с постоянными параметрами, выбираемые так, чтобы пара (Λ^{τ}, H_i), $i = 1, 2$ была управляемой; $\Gamma \in R^{(2m-1)\times(2m-1)}$ — диагональная матрица с положительными диагональными членами.

Вектор $E(t)$ характеризует отклонение движения системы с АН от опорного движения $(\hat{x}^*(t), x^*(t), A_3)$. При этом предполагается, что $\hat{x}^*(t_0) = x^*(t_0)$.

Задача анализа стохастической устойчивости системы с АН сводится к определению условий, при которых является стохастически устойчивым тривиальное решение системы (2) $E(t) = 0$.

В зависимости от функции $\beta(t)$, входящей в вектор D, будем рассматривать следующие два частных вида системы (2). Систему (2) с функцией $\beta(t) = \tilde{\delta} - \Delta a_1(t)$ $(\tilde{\delta} = \lambda_1 - a_1;\ \Delta a_1 = \hat{a}_1 - a_1)$ будем обозначать через S_1, а систему (2) с функцией $\beta(t) = \lambda_1 + \hat{a}_1(t)$ — через S_2. Система S_2 является стохастическим аналогом системы с АН, предложенным в [7].

Найдем ограничения на интенсивность действующей помехи η (ξ^u, как видно из (2), не оказывает влияния на работу системы), при которых гарантируется устойчивость тривиального решения S_1, S_2 — систем при постоянно действующих случайных возмущениях, малых в среднем квадратичном. На движениях системы (2) будем рассматривать две функции Ляпунова: $V_1(t) = \frac{1}{2}\,e_2^1(t)$; $V_2(t) = \frac{1}{2}\Delta A_3^{\tau}(t)\,\Gamma^{-1}\Delta A_3(t)$. Производящие дифференциальные операторы для V_1, V_2 определяются формулами

$$L V_1 = (de_1, \frac{\partial}{\partial e_1}) V_1 + \frac{1}{2} \sigma^2 y (\beta, \frac{\partial}{\partial e_1})^2 V_1,$$

$$L V_2 = (d\Delta A_3, \frac{\partial}{\partial \Delta A_3}) V_2 + \frac{1}{2} \sigma^2 y (D_1, \frac{\partial}{\partial \Delta A_3})^2 V_2,$$

где $D_1 = -\Gamma \tilde{S} e_1$

Будем считать, что выполняются следующие условия:

1) для $u(t)$ справедливо условие D_0;

2) матрица Λ является гурвицевой, $\lambda_1 > 0$ — некоторое число;

3) положительно определенные функции V_1, V_2 допускают бесконечно малый высший предел и

$$g_1 \| \Delta A_3 \|^2 \leqslant V_2 \leqslant g_2 \| \Delta A_3 \|^2,$$

где g_1, g_2 — соответственно наименьшее и наибольшее собственные числа матрицы Γ^{-1};

4) $\tilde{\delta} \leqslant \mu$, μ — достаточно малое положительное число;

5) $\| Z(t) \| \leqslant \tilde{z} < \infty$, $\forall t \in [t_0, \infty)$, $\Delta a_1^2 \leqslant \kappa V_1$ κ — некоторая положительная константа, $\| \cdot \|$ — евклидова норма.

Пусть для $\forall t \in [t_0; \infty)$ справедливы оценки

$$M \{V_i(t)\} \leqslant \rho_i(t) \quad i = 1, 2,$$

если $M \{V_i(t_0)\} \leqslant \rho_i(t_0)$.

Тогда для $L V_1$, $L V_2$, определенных на движениях S_1 — системы, можно построить матричную систему сравнения |СС|.

Лемма. Невозмущенное движение матричной СС

$$\begin{bmatrix} \dot{\rho}_1 \\ \dot{\rho}_2 \end{bmatrix} = \begin{bmatrix} -\lambda & \frac{1}{2}(\frac{\tilde{z}^2}{\lambda_1 g_1} + \kappa \sigma^2 y) \\ \frac{1}{2} \gamma_1 \sigma^2 y & -\tilde{z}^2 (\gamma_1 g_2 \sigma^2 y)^{-1} \end{bmatrix} \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} \tag{3}$$

при выполнении условий 1) — 5) будут устойчиво, если

$$\sigma^2 y \leqslant \frac{\lambda_1 \tilde{z} \sqrt{3 g_1}^3 \sqrt{12 \pi^2 \tilde{z}}}{\{\gamma_1 [9 \pi^2 \gamma_1 g_2 \tilde{z}^2 (\kappa g_1 \lambda_1^2 \sqrt{3 g_1} + \gamma_1 \tilde{z}^2 \sqrt{g_2}) + 2 \sqrt{g_2} (27 \kappa^2 g_1^3 \lambda_1^4 - \gamma_1^2 g_2 \tilde{z}^4)]^{1/3}} =$$

$$= \Theta, \tag{4}$$

где γ_1 — первый диагональный элемент матрицы Γ.

Доказательство леммы основано на применении леммы 2 из [6]. Если в (3) положить $\kappa = 0$, то получим невозмущенную матричную СС для S_2 — системы, решение которой будет устойчиво при соблюдении следующего ограничения

$$\sigma^2 y \leqslant 2 \frac{\lambda_1}{\gamma_1} \sqrt{\frac{g_1}{g_2}}$$

Теорема 1 [6]. Пусть выполняются условия 1) — 5), а также:
6) производящие операторы функций V_1, V_2, удовлетворяют неравенствам

$$L V_1 \leqslant -\lambda_1 V_1 + \frac{1}{2}\left(\frac{\tilde{z}^2}{\lambda_1 g_1} + \kappa\sigma^2 y\right) V_2 + \frac{1}{2}\mu\sigma^2 y,$$

$$L V_2 \leqslant -\frac{\tilde{z}^2}{\gamma_1 g_2 \sigma^2 y} V_2 + \frac{1}{2}\gamma_1 \sigma^2 y V_1;$$

7) $\sup \sigma^2 y \leqslant \theta, \forall t \in [t_0, \infty)$, где θ определяется (4).
Тогда тривиальное решение S_1 — системы устойчиво при постоянно действующих случайных возмущениях, малых в среднем квадратичном, и

$$\lim_{t \to \infty} M\left\{\|E(t)\|^2\right\} \leqslant \mu\tilde{\beta} \sup_t \sigma^2 y,$$

где $\tilde{\beta}$ — некоторое положительное число.

Теорема 2 [6]. Пусть выполняется условие теоремы 1, кроме условия 4) где $\mu = 0$. Тогда тривиальное решение S_1 — системы асимптотически устойчиво по вероятности с оценкой

$$M\left\{V_1(e_1(t))\right\} \leqslant e^{-\alpha_1 (t - t_0)}\left\{\nu_1 V_1(e_1(t_0)) + \nu_2 V_2(\Delta A_3(t_0))\right\},$$

где ν_1, ν_2 — положительные числа; $\alpha_1 > 0$ — число, определяющее степень устойчивости матричной СС (3).

Следствие. Пусть выполняются условия теоремы 2. Тогда тривиальное решение S_1 — системы экспоненциально устойчиво в среднем квадратичном с оценкой

$$M\left\{\|E(t)\|^2\right\} \leqslant k\|E(t_0)\|^2 e^{-\alpha_1 (t - t_0)} \quad \forall t \in [t_0, \infty),$$

где k — положительное число.

Теоремы 1, 2 устанавливают область устойчивости S_1 — системы. Интенсивность помехи ξ^y ограничена сверху величиной (4), которая зависит от минимального и максимального собственных чисел матрицы Г, нормы вектора Z и параметра λ_1. Величина λ_1 должна выбираться как можно ближе к первому элементу вектора A_3, что соответствует $\mu \to 0$, так как в противном случае дисперсия ошибки $E(t)$ будет увеличиваться. Если λ_1 совпадает с a_1, то гарантируется экспоненциальная устойчивость по отношению к помехе измерения выхода объекта (1).

К сожалению, оценкой (4) для $\sigma^2 y$ в практических приложениях пользоваться затруднительно. Более простую, но несколько завышенную оценку для $\sigma^2 y$ можно получить, исходя из следующего предположения $\|Z\|^2 \geqslant \rho\sigma^2 y$, ($\rho > 0$ — некоторая константа). Тогда

$$\sup \sigma^2 y \leqslant \sqrt[3]{\frac{4\lambda_1^2 \tilde{z}^2 g_1}{\gamma_1^2 g_2 (\rho + \kappa\lambda_1 g_1)}} \qquad \forall t \in [t_0, \infty).$$

Для S_2 — системы справедлива

Теорема 3 [6]. Пусть выполняются условия 1) — 3), 5), а также:
8) производящие операторы функций V_1, V_2 удовлетворяют неравенствам

$$LV_1 \leqslant -\lambda_1 V_1 + \frac{1}{2}\frac{\tilde{z}^2}{\lambda_1 g_1} V_2 + \frac{1}{2}(\lambda_1 + \hat{a}_1)^2 \sigma^2 y;$$

$$LV_2 \leqslant -\frac{\tilde{z}^2}{\gamma_1 g_2 \sigma^2 y} V_2 + \frac{1}{2}\gamma_1 \sigma^2 y\, V_1;$$

9) интенсивность помехи удовлетворяет неравенству

$$\sup_t \sigma^2 y \leqslant 2\frac{\lambda_1}{\gamma_1}\sqrt{\frac{g_1}{g_2}} \qquad \forall t \in [t_0, \infty).$$

Тогда S_2 — система диссипативна и

$$\lim_{t \to \infty} M\left\{\|E(t)\|^2\right\} \leqslant c(\lambda_1 + \hat{a}_1)^2 \sup_t \sigma^2 y,$$

где $c > 0$ — некоторая постоянная.

Доказательство полученных результатов приведено в [6].

ЛИТЕРАТУРА

1. Landau J.M. Martingale conwergence analisis of adaptive schemes — a feedback approach. — IEEE Trans. Autom. Control, 1982, V. AC — 27, № 3, pp. 716—719.
2. Dugard L., Landau J.M., Silveira H. Adaptive srable estimation using MRAS techniques — convergence, analisis evaluation. — IEEE. Trans. Autom. Control, 1980, V. AC — 25, № 6, pp. 1169—1182.
3. *Наредра К.С., Валавани Л.С.* Устойчивые адаптивные наблюдения и управления. — ТИИЭР, 1976, т. 64, № 8, с. 94—106.
4. Lion P.M. Rapid identification of linear and nonlinear systems. — AIAA Sournal, vol. 5, oct. 1967.
5. *Ядыкин И.Б.* О стохастической устойчивости одного класса беспоисковых адаптивных систем управления с эталонной моделью. Автоматика и телемеханика, 1981, № 3, с. 56—68.
6. *Карабутов Н.Н.* Исследование стохастической устойчивости адаптивных наблюдателей неявного типа. — Рукопись депон. В ВИНИТИ, 1985, № 2804—85 Den., 29 с.
7. Lüders G., Narendra K.S. An adaptive observer and identifier a linear system. — IEEE. Trans. Autom. Control, 1973, V. AC — 18, № 5., 496—499.

Adaptiv megfigyelők sztochasztikus stabilitása vektor Ljapunov függvények segitségével

N.N. Karabutov
I. Hadrevi

Összefoglaló

A cikkben az implicit tipusu lineáris dinamikus objektumok adaptiv megfigyelőinek sztochasztikus stabilitásáról van szó. Az egyenleteket a szerzők egy nem-minimális identifikációs formára hozzák, amelyeket aztán a vektor Ljapunov függvény módszer sztochasztikus variánsával vizsgálják.

Stochastic stability of adaptive observers using vector Ljapunov functions

N.N. Karabutov
I. Hadrevi

Summary

In the paper stochastic stability of adaptive observers of linear dynamic objects of implicite-type is investigated. Their equations are transformed to non-minimal identification form using the stochastic variant of the method of vector Ljapunov functions.

MTA SZTAKI Közlemények 33/1985 p. 37-45

AUTOMATED PROTOCOL VERIFICATION

László KOVÁCS

Computer Network Department
Computer and Automation Institute of the
Hungarian Academy of Sciences

1. INTRODUCTION

Distributed systems and computer networks are widely used all over the world. A protocol represents a set of rules which control the inner communication of a computer network. The correct operation of this algorithm fundamentally influences the operation and the realiability of the whole computer network. The use of an erroneous protocol can result in incalculable consequence so that at present the protocol designers specify protocols with great care. Formal techniques to check the protocol correctness are called verification methods.

The precondition of a protocol verification is the existence of the formal description of the protocol and its communication services. In this paper we present an automated protocol verification method which analyses all possible behaviour of a protocol system described in a high level specification language. This verification method is one of the family of "global view" methods because of its use of global state information. The "local view" methods (e.g. program proving approach) are based on the analysis of the interactions between the protocol entities and their local encironments. [Bochmann-Sunshine 80]

2. PROTOCOL PROPERTIES

Verifiable protocol properties may be classified into two categories: ([Sunshine 79]) specific and general properties. Specific properties are those that characterize the protocol individually. The most important specific protocol property is that of the functional correctness. It means that the protocol meets its service specification. The general properties are those that are common to all protocols. The author considers that the properties described below are the most essential general properties.

The completeness of the protocol means that the protocol entities detect all interactions arrived from their environments and produce output interactions when required.

Protocols are expected to operate free from deadlock. Deadlock is a system state from which there is no exit. In the case of deadlock there is a logical loop of protocol entities in which each entity is waiting for interaction of the preceding one.

The liveness property means that every protocol system state is reachable from each reachable system state.

The progress (or tempo-blocking freeness) protocol property is the absence of cyclic behaviour where no useful activity takes place. In other words the protocol never gets into any non-productive looping.

The recoverability concept is based on the view of dividing the protocol system state space into two parts: the normal and the abnormal spaces. The protocol is recoverable if after the occurence of a temporary aberration the protocol will return to the normal operation space in a finite time [Merlin-Farber 76].

The protocol is stable if the step's number to recover the protocol does not increase in the operation time.

3. ABSTRACT MODEL OF PROTOCOLS

3.1. THE STATE TRANSITION APPROACH

We make a proposal as to the abstract conceptual model of protocol systems. To model the protocols we present a non-deterministic partitioned named transition system which is a modified version of the Keller model of parallel computation [Keller 76].

Definition 1. A partitional transition system is a quadrople (Q,R,N,P) where Q is a set of states, R is a binary relation on Q, N is a set of (action) names and P is a binary partition of actions.
$(p \in P \wedge \forall i<>j(p[i] \cap p[j] = \emptyset) \wedge \forall i(p[i] <> \emptyset))$ @

Denote $q \rightarrow q'$ the R relation of $q,q' \in Q$ states which is called state transition. The system in the case of $q \rightarrow q'$ executes an indivisible (atomic) action. (Notation:

$q \xrightarrow{p.n} q'$, $n \in N$, $p \in P$)

Definition 2. Let (Q,R,N,P) be a partitional transition system. The set of possible next states is defined to be
$E(q) = (q' : q \rightarrow q' \wedge q,q' \in Q)$ @

Definition 3. The set of enabled actions is defined to be

$V(q) = \{n : n \in N \wedge q' \in E(q) \wedge q \xrightarrow{p.n} q'\}$ @

Definition 4. The $S = q[0], q[1],\ldots q[i],\ldots$ is a state sequence of the (Q,R,N,P) system if and only if
$\forall i\ (q[i] \in Q \wedge i > 0 (q[i] \in E(q[i-1])))$ @

Definition 5. The $T = n[1], n[2],\ldots n[i], \ldots$ is computation on the (Q,R,N,P) system if and only if
$\exists S\ (\forall i\ (n[i] \in V(q[i-1])))$ @

Definition 6. The (Q,R,N,P) system is nondeterministic if $\exists q\ (q \in Q \wedge \#E(q)>1)$ @
(Notation: # cardinality)

In protocol systems there are parallel overlapped actions. To describe protocol we apply the nondeterministic (Q,R,N,P) system so that we transform the parallel systems into that kind of system. In the transformation we represent the parallel overlapped actions with their sequential execution of optional order.

Definition 7. The total protocol specification is the definition of the (Q,R,N,P) transition system. @

To form a method which can be used in practice let K be a predicate and let L be a state transition function on objects representing Q space. The correspondence between the (K,L) and the (Q,R,N,P) systems is the following.

$q,q' \in Q,\ p \in P,\ n \in N\ ((q \xrightarrow{p.n} q')$ if and only if $(K(p\cdot n,REP(q)) \wedge REP(q') = L(p.n,REP(q))))$.
The REP function refers to the objects representing Q space.

Definition 8. The protocol specification is the description of the (K,L) representation. @

The main difference between the two definitions (7. and 8.) is that (K,L) is a "human executable" and (Q,R,N,P) is a "machine executable" model. In the [Kovacs 82] report the author published a high-level language implementation of the (K,L) system.

3.2. FORMULATIONS OF PROTOCOL PROPERTIES

In order to formulate the protocol properties presented in the second section of this paper define the concept of the reachability relation (R^*). Let R^* be the reflexive and transitive closure of R (Keller definition 3.1). (Notation: $\xrightarrow{*}$)

Definition [Keller definition 3.2) *9.* Let I be a predicate on Q. I q[O]-invariant in (Q,R,N,P) if and only if $\forall$ q (q[O] $\xrightarrow{*}$ q $\wedge$ I(q)) @ (q[O] is the start state of the system).

In the formal description of deadlock freeness protocol property we follow the extention of the Keller's definition to the (Q,R,N,P) model.

Definition 10. Let ACTIV be a predicate on Q. It is defined to be ACTIV(q) if and only if #E(q))>= 1 @

Definition 11. The protocol is deadlock free if and only if ACTIV(q) q[O]-invariant @

Formulation of the liveness property corresponds to the informal description of the 2. section.

Definition 12. Let REACHABLE be a predicate on Q. REACHABLEq'(q) if and only if q $\xrightarrow{*}$ q' @

Definition 13. The protocol has liveness property if and only if (REACHABLEq q[O]-invariant) q[O]-invariant. @

In this formalism it appears to be very difficult to construct a formal definition of the progress property since it requires the formulation of the usefulness of the protocol cycles so we do not try it.

4. PROTOCOL VERIFICATION

4.1. IMPROVED REACHABILITY ANALYSIS

The improved reachability analysis verification method check the q[O]-invariance of a predicate. It is based on the reconstruction of (Q,R,N,P) transition system from (K,L) protocol representation (language model). This reconstruction consists of building a G graph of (Q,R,N,P) system. By the G graph of (Q,R,N,P) system we mean an oriented graph whose

nodes correspond to the Q states and whose arrows correspond to the state transitions labelled by action names. In other words this means that there is an arrow from the node indicated by q to the node q' labelled by p·n if and only if there exists $q \xrightarrow{p \cdot n} q'$ relation in (Q,R,N,P) system. The correspondence between the structural (geometrical) properties of the G graph and the protocol properties defined in the 3.2 section makes it possible to map the protocol verification to the structural analysis of the G graph.

The node in the G graph from which no arrows descend represents a protocol deadlock state (definition 11.). Protocol has liveness property if the G graph is strongly connected (definition 13.). Several algorithms were published to examine the strong connectivity of an oriented graph (e.g. [Aho-Hopcroft-Ullman 75]). The protocol cycles correspond to the loops of G graph. Examination of protocol cycles to determine the useless loops (progress property) concerning the human designers demands the knowledge of a concrete algorithm. Other protocol properties formulating in the form of predicate q[O]-invariance can be examined with the evaluation of predicates in some nodes of G graph.

In order to put this method into practice we have to define the system state in the case of concrete language representation of (K,L) system. It means the definition of the REP^{-1} function. The language representation of (K,L) system was published in [Kovacs 82] uses entities communicating with each other to describe the protocol. In that case the REP^{-1} function was defined a $q = \langle e[1], e[2], \ldots e[n] \rangle$ where q is equal to a vector (n-tuple) of the states of entities.

4.2. COMPUTER-AIDED VERIFICATION OF PROTOCOLS

The automated protocol verification system described below is an interactive software tool realizing the improved reachability analysis of 4.1 section. In the first step the

protocol designer describes the protocol by means of the specification language. In that process the designer uses the language constructions and abstract data types predefined in the open abstract data type library of the system. The specification language makes it possible to express the protocol properties demanded by the designer in the form of predicates. The language specification of the protocol is the input of the verification system. In the first pass the system compiles the specification then it starts the construction of G graph and the examination of it's structural properties. The designer can visualize the system informations in the way of the progress of the verification by means of a display. The job of designer sitting before the display is to "cut" the infinite branch of G graph in the case when no predicate signals it. The user of the system can stop the verification process by an interactive command and can ask the actual values of protocol variables.

The speed of system operation is determined to the greatest extent by the fact that the system can't keep the whole G graph in memory because of it's size. It is necessary to swap the parts of the graph to the disk. As a result of this fact, a protocol verification process exceeds the average session time.

The first experiences of verifying system operation - despite the awkwardness of the system - strengthen our resolution which can lead to the interactive automated systems for analyzing and synthesizing protocols in the near future.

REFERENCES

[Aho-Hopcroft-Ullman 75] Aho V.A., Hopcroft J.E., Ullman J.D., "The Design and Analysis of Computer Algorithms," Addison-Wesley Publ. Comp. 1975.

[Bochmann-Sunshine 80] Bochmann G.V., Sunshine C.A., "Formal Methods in Communication Protocol Design," IEEE Trans. on Comm. Vol. COM-28, No. 4. Apr. 1980.

[Keller 76] Keller R.M., "Formal Verification of Parallel Programs," Comm. of the ACM Vol. 19. No. 7. Jul. 1976.

[Kovacs 82] Kovacs L., "Formal Specification and Verification of Computer Network Protocols," Technical Report, MTA SZTAKI Tanulmányok 138/1982. (In Hungarian)

[Merlin-Farber 76] Merlin Ph.M., Farber, D.J., "Recoverability of Communication Protocols - Implications of a Theoretical Study," IEEE Trans. on Comm. Vol. COM-24, Sep. 1976.

[Schultz-Rose-West-Gray] Schultz G.D., Rose D.B., West C.H., Gray J.P., "Executable Description and Validation of SNA," IEEE Trans. on Comm. Vol. COM-28, No. 4. Apr. 1980.

[Schwabe 81] Schwabe D., "Formal Techniques for the Specification and Verification of Protocols," Technical Report, UCLA Apr. 1981.

[Sidhu 82] Sidhu D.P., "Rules for Synthesizing Correct Communication Protocols," Comp. Comm. Rev. Vol. 12. No. 1. Jan. 1982.

[Sunshine 79] Sunshine C.A., "Formal Techniques for Protocol Specification and Verification," Computer, Sep. 1979.

Automatikus protokoll verifikálás

Kovács László

Összefoglaló

A dolgozat a számitógép-hálózati protokollok verifikálására, helyességének ellenőrzésére szolgáló automatikus módszert mutat be. Definiálja a protokollok egy absztrakt matematikai modelljét. Az absztrakt modell lehetőséget teremt a protokollok legfontosabb tulajdonságainak formális értelmezésére. A közölt módszer a protokollok specifikációs nyelvü modelljét visszavezeti az absztrakt modellre, tehát az implementált verifikáló rendszer képes a protokollok tulajdonságait felderiteni.

АВТОМАТИЧЕСКАЯ ВЕРИФИКАЦИЯ ПРОТОКОЛОВ

Л. Ковач

Р е з ю м е

В статье описывается автоматический метод проверки правильности и верификации протоколов вычислительных сетей. Определяется абстрактная математическая модель протоколов. Абстрактная модель дает возможность для формального толкования важнейших характеристик протоколов. Описанный метод сводит модель со спецификационным языком к абстрактной модели протокола, следовательно реализованная система верификации способна определить характеристики протоколов.

MTA SZTAKI Közlemények 33/1985 p. 47-54

LA NON-STABILITÉ NEGATIVE IMPLIQUE DES PÉRIODES DOUBLES

NGUYEN CONG THANH

Institut de Recherch Météorologique et
Hydrologique
Hanoi

En considerant un système dynamique on voit que la non--stabilité des orbites périodiques en générale conduit au changement du comportement asymptotique de système. C'est le phénomène de bifurcation. Les bifurcation locales typiques dans le cas d'une dimension sont exprimmées dans les théorèmes bien connus de Guckenheimer (2). Dans les bifurcations possibles on s'interesse bien aux bifurcations de période double. Puisque le phénomène de bifurcations de période double est utilisé pour expliquer l'une des manières de naissance de la turbulence. Autrement dit, la turbulence dans quelques systèmes peut être considerée comme l'accumulation de successives bifurcations de période double (voir [1],[3]). Le théoreme suivant montre que le phénomène de bifurcations de periode double est la consequence de la non-stabilité négative. C-à-d. la non-stabilité négative des points périodeques de période n d'un système à une dimension implique l'existence des points périodiques de période 2N.

THEOREME

Soit f une application de classe C^1 de l'intervalle $I = [0,1]$ dans lui-même. S'il existe un point fixe non--stable négatif, c-à-d. il existe un point $z \in I$ tel que $f(z) = z$ et $f'(z) < -1$. Alors l'application f a une orbite de periode 2.

Dans la demonstration du theorême on utilise la suivante

Proposition 1.

Si f satisfait aux conditions du théorème, il existe un voisinage $U = (z+\varepsilon, z+\varepsilon)$ de u tel que pour tout x U on ait toujours des entiers $n(x), m(x)$ pour lesquels

$$f^{n(x)}(x) - z > \varepsilon; \quad f^{m(x)}(x) - z < -\varepsilon$$

Demonstration de la Proposition 1

Soit $f'(z) = -C$, ou $C > 1$, par la continuté de f' il existe un voisinage $V = (z-\delta, z+\delta)$ de z tel que

$$-\sqrt{C} > f'(x) > C_o^2 \quad \text{pour tout } x \in V$$

Nous considerons le voisinage $U = (z-\varepsilon, z+\varepsilon)$ avec $\varepsilon = \delta/c^3$, Quand $x \in U$ on a toujours $|f'(x)| > \sqrt{C} > 1$ donc il existe un n_1 tel que $f^{n_1}(x)$ ne soit pas contenu dans U. On note

$$n = \min \{n_1 \text{ tel que } f^{n_1}(x) \notin U\}$$

Alors $|f^n(x) - z| > \varepsilon$ et $|f^i(x) - z| < \varepsilon \ \forall\ i < n$.

Sans limiter la généralité on peut supposer que

$$f^n(x) - z > \varepsilon \tag{1}$$

Alors

$$|f^n(x) - z| = |f \circ f^{n-1}(x) - z| = |f'(\eta)| \cdot |f^{n-1}(x) - z| \tag{2}$$

où η est une valeur entre $f^{n-1}(x)$ et z. Puisque $f^{n-1}(x)$ reste encore dans le voisinage U, η reste aussi dans ce voisinage, donc $f'(\eta) > C^2$.

En remplacant dans (2) on a

$|f^n(x) - z| < C^2 \cdot \varepsilon < C^2$. $\delta/c^3 < \delta$, c-à-d. $f^n(x) \in V$

Nous considerons la difference $f^{n-1}(x) - z$ et avons

$$f^{n+1}(x) - z = fof^{n}(x) - z = f'(\nu)\cdot(f^{n}(x)-z)$$

ou est entre $f^{n}(x)$ et z donc il reste encore dans le voisinage V.

Alors

$$f'(\nu)\cdot(f^{n}(x) - z) < \sqrt{C}\cdot\varepsilon < \varepsilon \quad \text{c-à-d. } f^{n+1}(x) - z < \varepsilon$$

La démonstration de la Proposition 1 est terminée.

Remarque

De cette Proposition on peut deduire que dans ce voisinage il y a des points t, pour lesquels $f(t) > t$ (resp. $f(t) < t$) et il existe un certain entier n tel que $f^{n}(t) < t$ (resp. $f^{n}(t) > t$). En effet, si $t \in U$ suppose $f(t) < t$, on a $z+\varepsilon > t$ et d'après la Proposition il existe n tel que

$$f^{n}(t) - z > \varepsilon \quad \text{Donc } f^{n}(t) > z+\varepsilon > t.$$

Démonstration du Théorème

Pour tout $x \in I$ on definit un nombre entier $n(x)$ comme suivant

si $f(x) > x$, $n(x)$ tel que $f^{n}(x) > x$ avec $n \leq n(x)$ et

$$f^{n(x)+1}(x) \leq x$$

si $f(x) < x$, $n(x)$ tel que $f^{n}(x)$ x avec $n \leq n(x)$ et

$$f^{n(x)+1}(x) \geq x$$

et pose

$$N = \min\{n(x), \forall\, x \in I\}$$

d'après la Proposition précédente N est un nombre entier fini. Dans le cas $N = 1$ la conclusion du Théorème est donnée dans la

Proposition 2

Soit f une application continue de I dans lui-même. Suppose f ait des points $x \in I$ avec les proprietés suivantes

$$f(x) > x \text{ (resp. } f(x) < x) \text{ et } f^2(x) \leq x \text{ (resp. } f^2(x) \geq x)$$

Alors il existe une orbite périodique de periode 2.

Nous ferons la démonstration dans le cas $f(x) > x$ et $f^2(x) \leq x$.

Vu que $f^2(x) < x$ et $f^2(0) \geq 0$, dans l'intervalle $[0,x)$ is existe au moins un point y tel que $f^2(y) = y$. On appelle p le point le plus proche de x tel que $f^2(p) = p$. Alors $f^2(q) \neq 1$ avec $q \in (p,x)$.
Puisque $f^2(x) < x$ on a

$$f^2(q) < q \quad \text{avec tout} \quad q \in (p,x) \tag{1}$$

Si p n'était pas un point d'une orbite périodique de periode 2, alors p serait un point fixe de f : $f(p) = p$. Dans l'intervalle (p,x) on aurait toujours $f(q) \neq q$ et $f(x) > x$ donc

$$f(q) > q \quad \text{avec tout} \quad q \in (p,x) \tag{2}$$

Alors il existerait un point t près de p tel que $t \in (p,x)$

$$p < f(t) < x \tag{3}$$

Par (2) et (3) on aurait $f^2(t) = fof(t) > f(t) > t$. Celui est avec (1). Donc $f(p) \neq p$, et p est un point périodique de periode 2.

Le cas où $f(x) > x$ et $f^2(x) = x$ est evident.

Nous revenons à la démonstration du Théorême dans le cas où $N \geq 2$. On note u le point, pour lequel $n(u) = N$ et suppose $f(u) > u$. Alors on a

$$f(u) > u,\ f^N(u) > u \quad \text{et} \quad f^{N+1}(u) < u$$

Note $v = f^N(u)$ on a $v > u$

En comparant v avec $f(u)$ on obtient les suivants

Dans le cas $v = f(u)$, c-à-d. $f^{N+1}\ f(u) = f(u)$.
Si $N \geq 3$ et $f\ f(u) \neq f(u)$ c'est $f(u)$ qui est point periodique de periode plus grand que 2. D' après Théorème de Sharkovski ([5]) il exists des points périodiques de période 2. Donc nous ne considerons le cas où de ce que

$$f^{N-1} \circ f(u) = f(u)$$

il résulte

$$fof(u) = f(u)$$

Dans le cas $v > f(u)$ i.e. $f^{N-1}of(u)$ $f(u)$ d'après la définition de N on a

$$fof(u) > f(u)$$

En somme, dans le cas $v \geq f(u)$ on obtient

$$fof(u) \geq f(u)$$

De la définition de N, en appliquant pour $f(u)$ on a

$$f^N of(u) \geq f(u) \quad \text{et}$$

$$f(v) = f^{N+1}(u) = f^N of(u) \geq f(u) > u$$

Cette contradiction excluse la possibilité $v \geq f(u)$.
Donc nous avons

$$f(v) \leq u < v < f(u)$$

D'après le théorème des valeurs intermediaires il existe des points dans (u,v), auxquelles la fonction f prend la valeur v. On note w le point le plus proche de u tel que

$$f(w) = v$$

Puisque $f(u) > v$, dans l'intervalle (v,w) on a toujours

$$f(x) > v > x \quad \text{pour tout} \quad x \in (u,w) \tag{4}$$

On a aussi

$$fof(w) = f(v) \leq u \quad \text{et} \quad fof(u) > u \quad \text{puisque} \quad f(u) > u.$$

Alors il existe un point $s \in (u,w)$ tel que

$$fof(s) = s$$

Par (4) le point s n'est pas un point fixe.

C.Q.F.D.

Remarque

En général la non-stabilité positive des points périodiques de periode N n'implique pas l'existence des points périodiques de periode 2N. Par exemple

$$f(x) = x^a \quad \text{avec} \quad a > 1.$$

REFERENCES

[1] M.J. Feigenbaum, The transition to aperiodic behaviour in turbulent systems. Comm. Math. Phys. V.77 (1980).

[2] J. Guckenheimer, On the bifurcations of maps of the interval. Invent. Math. 39(1977).

[3] R.B. May, Simple mathematical models with very complicated dynamics. Nature 261. (1976).

[4] Nguyen Cong Thanh, Dissertation, Budapest 1985.

[5] A.N. Sharkovski, Coexistence of cycles of a continuous map of a line into itself. Ukranian Math. J. 16(1964).

A negativ instabilitás implikálja a 2 periódusu pálya létezését

Nguyen Cong Thanh

Összefoglaló

A dolgozatban a következő tételt bizonyitjuk be: legyen f a $[0,1]$ intervallum egy C^1 leképzése önmagába; ha f-nek létezik egy $z \in [0,1]$ fixpontja ugy, hogy $f'(z) < -1$, akkor az f leképzésnek van 2 periódusu pályája.

The negative unstability implies the existence of 2-period point

Nguyen Cong Thanh

Summary

In this paper the following theorem is proved: let f be a C^1-mapping of the interval $[0,1]$ into itself; if there exists a fixed point $z \in [0,1]$ of f such that $f'(z) < -1$, then the mapping f has 2-period orbit.

ОТРИЦАТЕЛЬНАЯ НЕУСТОЙЧИВОСТЬ ВЛЕЧЕТ ЗА СОБОЙ СУЩЕСТВОВАНИЕ ОРБИТЫ ПЕРИОДА 2

Нгуен Конг Тхан

Р е з ю м е

В работе доказывается следующая теорема: пусть f отображение класса C^1 интервала $[0,1]$ в себя; если существует неподвижная точка $z \in [0,1]$ такая, что $f'(z) < -1$, то тогда отображение f имеет орбиту периода 2.

MTA SZTAKI Közlemények 33/1985 p. 55-67

ON FUZZY AUTOMATA AND FUZZY GRAMMARS

K.G.Peeva

Center of Applied Mathematics
Sofia 1000,P.O.Box 384

ABSTRACT. Fuzzy grammars and fuzzy languages in connection with finite fuzzy acceptors are studied. Let A be a finite fuzzy acceptor and R(A) be the set of all words recognizable by A. It is proved that for each fuzzy regular grammar G_F generating the language $L(G_F)$ there exists a finite fuzzy acceptor A such that $R(A)=L(G_F)$ and vice versa.

The main results are about algorithmical decidability of ε-equivalence and ε-reduction by inputs. It is shown that the relation ε-closeness of matrices is invariant. On this base some properties of the ε-equivalence and ε-reduction are obtained and their application in syntactic pattern recognition are discussed.

1. ε-CLOSENESS OF MATRICES

In this section ε-closeness for matrices over a bounded chain is defined and studied. These algebraic results are necessary for the ε-equivalence and ε-reduction by input words which is the subject of section 3. The algebraic terminology is according to [3].

Let $\mathbb{L}=([0,1],\vee,\wedge,0,1)$ be a bounded chain [3] over the ordered set $[0,1]\subset\mathbb{R}$ with lower and upper bounds respectively 0 and 1 and operations $\vee$ and $\wedge$.

Let $A=(a_{ij})_{mxn}$ and $B=(b_{ij})_{nxp}$ be matrices over the bounded chain $\mathbb{L}$ with elements $a_{ij},b_{ij}\in[0,1]$ for each i,j. The matrix $C=AB=(c_{ij})_{mxp}$ is a *product* of A and B if

$c_{ij} = \overset{n}{\underset{k=1}{\vee}} (a_{ik} \wedge b_{kj})$ for each i=1,...,m and each j=1,...,p [4].
It is easily established that the matrix multiplication is associative. Having in mind this property we shall omit the breckets next.

Let $A=(a_{ij})$ and $B=(b_{ij})$ be mxn-matrices and $\varepsilon \in [0,1]$ be fixed. We say that the i^{th} row in A is ε-*close* to the k^{th} row in B if $|a_{is}-b_{ks}| \leq \varepsilon$ holds for each s, $1 \leq s \leq n$; A and B are ε-*close* (notation $d(A,B) \leq \varepsilon$) if $|a_{ij}-b_{ij}| \leq \varepsilon$ holds for each i, $1 \leq i \leq m$ and for each j, $1 \leq j \leq n$.

Theorem 1. If $A=(a_j)_{nx1}$ and $B=(b_j)_{nx1}$ are ε-close then $|\overset{\vee}{j}(a_j)-\overset{\vee}{j}(b_j)| \leq \varepsilon$ is valid.

Proof. Let $\overset{\vee}{j}(a_j)=a_k$ and $\overset{\vee}{j}(b_j)=b_r$. Then

$a_k-\varepsilon \leq b_k \leq b_r \leq a_r+\varepsilon \leq a_k+\varepsilon \Rightarrow$
$a_k-\varepsilon \leq b_r \leq a_k+\varepsilon \Rightarrow$
$-\varepsilon \leq b_r-a_k \leq \varepsilon \Leftrightarrow |b_r-a_k| \leq \varepsilon$, i.e. $|\overset{\vee}{j}(a_j)-\overset{\vee}{j}(b_j)| \leq \varepsilon$

In particular, if $|a_k-a_i| \geq 2\varepsilon$ for each $i \neq k$ then $\overset{\vee}{j}(a_j)=a_k$ and $\overset{\vee}{j}(b_j)=b_k$ for the same index k because

$a_i+2\varepsilon \leq a_k \Leftrightarrow a_i+\varepsilon \leq a_k-\varepsilon \Rightarrow b_i \leq a_i+\varepsilon \leq a_k-\varepsilon \leq b_k$ and hence $b_i \leq b_k$ for each $i \neq k$.

It is easy to see that Th.1 is valid for A^t and B^t as well.

Theorem 2. If $d(A,B) \leq \varepsilon$ and $d(C,D) \leq \varepsilon$ then $d(AC,BD) \leq \varepsilon$ holds whenever the products make sence.

Proof. According to the definitions $d(AC,BD) \overset{\vee}{\leq} \varepsilon \Leftrightarrow$ $|\overset{\vee}{k}(\wedge(a_{ik},c_{kj}))-\overset{\vee}{k}(\wedge(b_{ik},d_{kj}))| \leq \varepsilon$ for each i,j. We shall prove the last inequation for arbitrary i,j. For the vector-matrices

$V_{ij} = (v_{ij}(k)) = (\wedge(a_{i1},c_{1j}),\wedge(a_{i2},c_{2j}),...)$
$W_{ij} = (w_{ij}(k)) = (\wedge(b_{i1},d_{1j}) \wedge(b_{i2},d_{2j}),...)$

we obtain $d(V_{ij},W_{ij}) \leq \varepsilon$ because $|v_{ij}(k)-w_{ij}(k)| \leq \varepsilon$ for each k as it is shown by the following points:

1. If $v_{ij}(k)=\hat{}(a_{ik},c_{kj})=a_{ik}$ and $w_{ij}(k)=\hat{}(b_{ik},d_{kj})=b_{ik}$ then $|v_{ij}(k)-w_{ij}(k)|=|a_{ik}-b_{ik}|\leq\varepsilon$ because $d(A,B)\leq\varepsilon$;

2. If $v_{ij}(k)=\hat{}(a_{ik},c_{kj})=c_{kj}$ and $w_{ij}(k)=\hat{}(b_{ik},d_{kj})=d_{kj}$ then $|v_{ij}(k)-w_{ij}(k)|=|c_{kj}-d_{kj}|\leq\varepsilon$ because $d(C,D)\leq\varepsilon$;

3. If $v_{ij}(k)=\hat{}(a_{ik},c_{kj})=a_{ik}$ and $w_{ij}(k)=\hat{}(b_{ik},d_{kj})=d_{kj}$ then $b_{ik}\in[a_{ik}-\varepsilon,a_{ik}+\varepsilon]\subset[0,1]$; $c_{kj}\in[a_{ik},1]$ because $a_{ik}\leq c_{kj}$. Since $d_{kj}\in[c_{kj}-\varepsilon,c_{kj}+\varepsilon]$ and $d_{kj}\in[0,a_{ik}+\varepsilon]$ $(d_{kj}\leq b_{ik}\leq a_{ik}+\varepsilon)$ we obtain $d_{kj}\in[a_{ik}-\varepsilon,a_{ik}+\varepsilon]$ and hence $|v_{ij}(k)-w_{ij}(k)|=|a_{ik}-d_{kj}|\leq\varepsilon$

4. If $v_{ij}(k)=\hat{}(a_{ik},c_{kj})=c_{kj}$ and $w_{ij}(k)=\hat{}(b_{ik},d_{kj})=b_{kj}$ by analogy with the previous case we obtain $|v_{ij}(k)-w_{ij}(k)|\leq\varepsilon$

Since $|v_{ij}(k)-w_{ij}(k)|\leq\varepsilon$ is valid for each k we have $d(V_{ij},W_{ij})\leq\varepsilon$. According to Th.1 the inequation $|\check{}_k\, v_{ij}(k)-\check{}_k\, w_{ij}(k)|\leq\varepsilon$ is true, i.e. $|\check{}_k(\hat{}(a_{ik},c_{kj}))-\check{}_k(\hat{}(b_{ik},d_{kj}))|\leq\varepsilon$ is valid.

Theorem 3. If $d(A,B)\leq\varepsilon$ then:

i) $d(CA,CB)\leq\varepsilon$; ii) $d(AT,BT)\leq\varepsilon$; iii) $d(CAT,CBT)\leq\varepsilon$ whenever the products make sence.

The proof follows from Th.2.

2. FUZZY ACCEPTORS AND FUZZY GRAMMARS

We define and study fuzzy acceptors and fuzzy grammars by analogy with [2] where the stochastic acceptors and stochastic grammars are considered. The terminology for automata and language theories is according to [2],[4],[5].

A fuzzy automaton A [4],[6] is a quintuple $A=(X,Q,Y,M,\mu)$ where:

(i) X,Q,Y are nonempty sets of input letters, states and output letters respectively;

(ii) $M=\{M(x/y)=(m_{ij}(x/y))/x\in X,y\in Y,m_{ij}\in[0,1]\}$ is the set of the transition-output matrices, the step-wise behaviour of A;

(iii) $\mathbb{L}=([0,1]^{\vee},\hat{}0,1)$ is the bounded chain.

If X,Q,Y are finite then A is called *finite automaton*.

The interpretation of the membership degrees $m_{ij}(x/y)\in[0,1]$ is well-known [4],[6]: each element $m_{ij}(x/y)$ determines the step-wise behaviour of A. If in step t the automaton is in state q_i and receives the input letter x, it puts out the output letter y in step t and reaches the state q_j in the next step t+1 with the membership degree $m_{ij}(x/y)\in[0,1]$.

A finite fuzzy acceptor (shortly *acceptor*) $A=(X,Q,q_o,F,M,\mathbb{L})$ is a finite fuzzy automaton without outputs (i.e. $|Y|=1$), with fixed initial state $q_o\in Q$ and with a set $F\subset Q$ of the final states.

Let X* be the free monoid generated by X with $e\in X^*$ as unit element. We extend the step-wise behaviour of the acceptor $A=(X,Q,q_o,F,M,\mathbb{L})$ to the complete behaviour of A for $k\in\mathbb{N}$ consecutive steps as follows: since the empty word e need no time we define $M(e)=I$, where I stands for the unitary matrix of order $|Q|$, the cardinality of Q; if the input word $u\in X^*$ is a letter $x\in X$ then the transition matrix is $M(u)=M(x)$; if in $k>1$ consecutive steps the letters $x_1,\ldots,x_k\in X$ are fed into A (i.e. the input word is $u=x_1\ldots x_k\in X^*$) then $M(u)=M(x_1)\ldots M(x_k)$ and an arbitrary element $m_{ij}(u)$ in M(u) is interpreted as the membership degree for the state q_i and the input word u in step t under teh state q_j in step t+k. We denote by M* the set of all transition matrices (the complete behaviour) for the given acceptor A:

$$M^* = \{M(u) = (m_{ij}(u))/u\in X^*\}.$$

The set of all words $u\in X^*$, which are recognizable by the acceptor A is denoted by R(A):

$$R(A) = \{u/u \in X^*, \exists m_{oj}(u)>0, q_j \in F\}.$$

We define the notion fuzzy grammar by analogy with [2] where stochastic grammars are defined and studied.

A fuzzy grammar $G_F=(N,T,S,P_F)$ is specified by a finite set N of nonterminal symbols, a finite set T of terminal symbols, disjoint from N, an element $S \in N$ called the start symbol and a finite set of fuzzy productions

$a_i \xrightarrow{p_{ij}} b_{ij}$, $i=1,\ldots,k$, $j=1,\ldots,n_i$, where $a_i \in (N \cup T)^* N (N \cup T)^*$, $b_{ij} \in (N \cup T)^*$ and $p_{ij} \in [0,1]$ is the mebership degree for this production. For the fuzzy grammar G_F we say that w *directly derives* w' (notation $w \xrightarrow{p_{ij}} w'$) with the membership degree p_{ij} iff $w=c_1 a_i c_2$, $w'=c_1 b_{ij} c_2$ and $a_i \xrightarrow{p_{ij}} b_{ij}$ is a production in P_F; we say that w *derives* w' with membership degree $p=\hat{}_j\, p_j$ (notation $w \overset{p}{\underset{*}{\rightarrow}} w'$) if there exists a sequence $w_1,\ldots,w_{n+1}$ in $(N \cup T)^*$ such that $w=w_1$, $w'=w_{n+1}$ and $w_j \xrightarrow{p_j} w_{j+1}$ for $1 \leq j \leq n$. The binary relation $\xrightarrow[*]{}$ is the reflexive and transitive closure of $\rightarrow$.

The *fuzzy language* $L(G_F)$ generated by G_F is the set of all terminal strings which can be derived from S:

$$L(G_F) = \{(u, p(u))/u \in T^*, S \overset{p_j}{\underset{*}{\rightarrow}} u, j=1,\ldots,k, p(u)=\overset{k}{\underset{j=1}{\vee}} p_j > 0\}.$$

The number of the different ways to obtain u from S is denoted by k.

Example 1. Let the fuzzy grammar $G_F=(N,T,S,P_F)$ with

$N = \{S_o, S_1, S_2\}$, $T=\{a,b\}$, $S=S_o$ and P_F:

$S_o \xrightarrow{0,1} aS_1$ $\quad S_1 \xrightarrow{0,2} aS_1$ $\quad S_1 \xrightarrow{0,7} a$

$S_o \xrightarrow{0,3} bS_2$ $\quad S_1 \xrightarrow{0,5} bS_2$ $\quad S_2 \xrightarrow{0,6} b$

be given. The fuzzy language $L(G_F)$ generated by G_F is

$$L(G_F)= \{(b^2,0,3)\} \cup \{(a^n b^2,0,1)/n\geq 1\} \cup \{(a^n,0,1)/n>1\}.$$

The grammar $G=(N,T,S,P)$ obtained from the fuzzy grammar $G_F=(N,T,S,P_F)$ by forgetting the membership degrees in the productions in P_F is called *associated* to G_F. The fuzzy grammar G_F has *type* 0,1,2,3 if its associated grammar G has type 0,1,2,3 respectively.

In this paper we consider only the fuzzy grammars of type 3. For a fuzzy grammar of type 3 (finite state, regular) all productions in P_F are as follows: $S_i \xrightarrow{p_{ij}} xS_j$ or $S_i \xrightarrow{p_i} y$, where $S_i,S_j \in N, x,y \in T$.

Theorem 4. Let $G_F=(N,T,S,P_F)$ be a fuzzy grammar of type 3. If $A=(X,Q,q_o,F,M,\mathbb{L})$ is an acceptor with $X=T$; $Q=N\cup\{E\}$; $q_o=S$; $F=\{S,E\}$ if the production $S \xrightarrow{p} e$ belongs to P_F and $F=\{E\}$ otherwise and the following membership degrees for each $q_i,q_j \in N$, $q_f \in F$, $x \in T$:

$m_{ij}(x)=p_{ij}>0$ if $q_i \xrightarrow{q_{ij}} xq_j$ is a production in P_F;

$m_{ij}(x)=p_i> 0$ if $q_i \xrightarrow{p_i} x$ is a production in P_F;

$m_{ij}(x)= 0$ otherwise,

then $R(A)=L(G_F)$.

Proof. Let $z_o=(z_o(i))_{1x|Q|}$ be the vector-row with elements

$$z_o(i) = \begin{cases} 1 & \text{if } q_i=q_o; \\ 0 & \text{if } q_i \neq q_o. \end{cases}$$

The product $z_o.M(u)$ determines the behaviour of A under the input word $u \in X^*$ if the initial state is $q_o \in Q$. Let $z_F=(z_F(i))_{|Q|x1}$ be the column-vector with elements

$$z_F(i) = \begin{cases} 1 & \text{if } q_i \in F, \\ 0 & \text{otherwise.} \end{cases}$$

Then $z_o.M(u).z_f \in [0,1]$ gives the maximal membership degree for the input word $u \in X^*$ if the beginning state is the initial state $q_o \in Q$ and the last state belongs to F. Obviously $u \in L(G_F) \iff z_o.M(u).z_F > 0 \iff u \in R(A)$.

Theorem 5. Let $A=(X,Q,q_o,F,M,\perp)$ be an acceptor. If $G_F=(N,T,S,P_F)$ is a fuzzy grammar with $N=Q$, $T=X$, $S=q_o$ and productions in P_F have the form $q_i \xrightarrow{p_{ij}} xq_j$ if $m_{ij}(x)=p_{ij}>0$, where $q_i,q_j \in Q$ and $x \in X$, or $q_i \xrightarrow{p_i} x$ if $m_{if}(x)=p_i>0$ for $q_i \in Q$, $q_f \in F$, $x \in X$, then G_F is a grammar of type 3 and $L(G_F)=R(A)$.

We may prove Th.5 in complete analogy with Th.4.

Example 2. Construct the acceptor, corresponding to the fuzzy grammar G_F given in Example 1.

According to Th.4 we obtain $X=\{a,b\}$, $Q=N \cup \{E\}=\{S_o,S_1,S_2,E\}$, $q_o=S_o$, $F=\{E\}$. The transition matrices are:

$$M(a) = \begin{pmatrix} 0 & 0{,}1 & 0 & 0 \\ 0 & 0{,}2 & 0 & 0{,}7 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad M(b) = \begin{pmatrix} 0 & 0 & 0{,}3 & 0 \\ 0 & 0 & 0{,}5 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The direct computation for $M(a^2)$ and $M(b^2)$ is:

$$M(a^2) = \begin{pmatrix} 0 & 0{,}1 & 0 & 0{,}1 \\ 0 & 0{,}2 & 0 & 0{,}2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad M(b^2) = \begin{pmatrix} 0 & 0 & 0 & 0{,}3 \\ 0 & 0 & 0 & 0{,}5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

By induction we can prove that

$$M(a^n) = \begin{pmatrix} 0 & 0{,}1 & 0 & 0{,}1 \\ 0 & 0{,}2 & 0 & 0{,}2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, n>1; \quad M(a^n b^2) = \begin{pmatrix} 0 & 0 & 0 & 0{,}1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} n \geq 1.$$

Since $z_o=(1\ \ 0\ \ 0\ \ 0)$ and $z_F^t = (0\ \ 0\ \ 0\ \ 1)$ we compute

$$z_o.M(b^2).z_F = 0,3;\ z_o.M(a^n).z_F = 0,1;\ z_o.M(a^nb^2).z_F=0,1.$$

The other words from X* are not acceptable since $z_o.M(u).z_F=0$. Hence this acceptor recognizes exactly the fuzzy language $L(G_F)$ from Example 1.

3. ε-EQUIVALENCE AND ε-REDUCTION

The classical problems for pure equivalence, reduction and minimization are completely studied for deterministic and non-deterministic automata [5]. Latter they were treated on the lines of their analogues for stochastic [5] and fuzzy [4],[6] automata.

Since the nature of stochastic and fuzzy automata is that they can be thought of as approximate models of incompletely understood systems the idea of approximate equivalence by (stochastic, resp. fuzzy) behaviours is well motivated.

We shall define and study the approximate equivalence and approximate reduction by inputs based on the ε-distance of the behaviour matrices for fuzzy acceptors. The main results concern the algorithmical decidability of the above problems.

The terminology on automata theory is according to [5].

Let $A=(X,Q,q_o,F,M,\mathbb{L})$ be an acceptor. The input words $u,v\in X^*$ are called *ε-equivalent* iff $d(M(u),M(v))\leq\varepsilon$ (notation $u \overset{\varepsilon}{\sim} v$).

Theorem 6. Let $A=(X,Q,q_o,F,M,\mathbb{L})$ be an acceptor, $x,x'\in X$ be input letters and $u,v\in X^*$ be input words. If $x \overset{\varepsilon}{\sim} x'$ then:

i) $uxv \overset{\varepsilon}{\sim} ux'v$;

ii) $xv \overset{\varepsilon}{\sim} x'v$;

iii) $ux \overset{\varepsilon}{\sim} ux'$.

Proof. $x \overset{\varepsilon}{\sim} x' \Longleftrightarrow d(M(x),M(x')) \leqq \varepsilon$. (i) follows from Th.3 (iii), (ii) follows from Th.3(ii);(iii) follows from Th.3(i)

Hence ε-equivalence by input letters implies ε-equivalence by input words, distinguished only by ε-equivalent letters (standing in the middle, in the beginning or in the end of the words).

The relation ε-equivalence by inputs is not an equivalence relation. But we can define an ε-partition on X, resp. on X*.

The set $[x_i]=\{x/x \in X \text{ and } x \overset{\varepsilon}{\sim} x_i\}$ defines an ε-*class* with *center* x_i. The ε-classes $([x_i])_i$ are called an ε-*partition* of X iff: $[x_i] \cap [x_j]=\emptyset$ for $i \neq j$ and $\bigcup_i [x_i]=X$. According to Th.6 the ε-partition on X induces an ε-partition on X*. Note that $[x_i] \neq [x_j] \Longrightarrow d(M(x_i),M(x_j))>\varepsilon$.

Theorem 7. For each acceptor $A=(X,Q,q_o,F,M,\mathbb{L})$ the following problems are algorithmically decidable:

i) whether $x \overset{\varepsilon}{\sim} x'$ for each $x,x' \in X$;

ii) construcing an ε-partition on X (resp. on X*).

Proof. i) For each $x,x' \in X$ we can compute whether $d(M(x), M(x')) \leqq \varepsilon$. The algorithm is finite because A is a finite acceptor. ii) We can construct an ε-partition on X using the following algorithm:

1. Enter X,M,ε.
2. For the elemenet $x_i \in X$ with the smalest index form the ε-class $[x_i]=\{x/x \in X \text{ and } x \overset{\varepsilon}{\sim} x_i\}$.
3. Print $[x_i]$.
4. $X = X - [x_i]$.
5. If $X \neq \emptyset$ go to Step 2.
6. End.

Let $A=(X,Q,q_o,F,M,\mathbb{L})$ and $A'=(X,Q',q'_o,F',M',\mathbb{L})$ be acceptors with the same $X,\mathbb{L}$. A and A' are *ε-equivalent by inputs* ($A \overset{\varepsilon}{\sim} A'$) if for each $x\in X$ there exists an ε-equivalent $x'\in X'$ (i.e. $d(M(x),M'(x'))\leq\varepsilon$) and vice versa. A' is in *ε-reduced form by inputs* if $x' \overset{\varepsilon}{\sim} x'' \Rightarrow x'=x''$ for each $x',x''\in X'$. A' is an *ε-reduct* of A if $A \overset{\varepsilon}{\sim} A'$ and A' is in ε-reduced form.

Theorem 8. Let $A \overset{\varepsilon}{\sim} A'$. For each input word $u\in X^*$ there exists an ε-equivalent input word $u'\in X'^*$ and vice versa.

Proof. If $u=e$ or $u\in X$ the proof is trivial. For $u=x_i x_j\in X$ the ε-equivalent word is $u'=x'x''\in X'^*$ if $x_i \overset{\varepsilon}{\sim} x'$ and $x_j \overset{\varepsilon}{\sim} x''$ because

$$x_i \overset{\varepsilon}{\sim} x' \Rightarrow d(M(x_i),M'(x'))\leq\varepsilon,$$

$$x_j \overset{\varepsilon}{\sim} x'' \Rightarrow d(M(x_j),M'(x''))\leq\varepsilon$$

and according to Th.2 $d(M(x_i),M'(x'))\leq\varepsilon$ and $d(M(x_j),M'(x''))\leq\varepsilon \Rightarrow d(M(x_i x_j),M'(x'x''))\leq\varepsilon$. The rest of the proof follows by induction on the lenght of the words and having in mind that $A \overset{\varepsilon}{\sim} A'$.

Corollary. If A' is an ε-reduct of A then A and A' have ε-equivalent behaviours.

Theorem 9. It is algorithmically decidable to find an ε-reduct for each acceptor $A=(X,Q,q_o,F,M,\mathbb{L})$.

Proof. According to Th.7 we can find an ε-partition X_r of X, where X_r is the set of the centers of the ε-classes. The different symbols in X_r are not ε-equivalent by construction. The acceptor $A_r=(X_r,Q,q_o,F,M_r,\mathbb{L})$ with $M_r=\{M(x)/x\in X_r\}$ is an ε-reduct of A.

4. APPLICATIONS IN SYNTACTIC PATTERN RECOGNITION

We shall sektch some applications of these results in syntactic pattern recognition. This is an open problem [1].

Let $W= \{(w,p(w))/p(w)\in[0,1]\}$ be a class of images and $p(w)$ is the membership degree for the image w. If each w is a string we can consider W as a fuzzy language $L(G_F)$. The set of the features P, wich characterizes each string of W, determines a finite set T of the terminals for the fuzzy grammar G_F, respectively the set X of the input letters for the recognizing acceptor A with $R(A)=L(G_F)$.

Let a suitable criterion with a numberical valuation $\varepsilon\in[0,1]$ be choosen, i.e. ε characterizes the similarity measure of the features in W. If the input letters $x,x'\in X$ are ε-equivalent, then $w=uxv$ and $w'=ux'v$ are ε-equivalent. But we can assign to each $x\in X$ a feature $p_x\in P$ and vice versa, i.e. $X\tilde{=}P$; consecuently the feature $p_x\in P$ is a carrier of an ε-equivalent information in comparison with $p_{x'}\in P$. Hence we can consider $p_{x'}$ as an inessential feature for the given recognizing problem or we can interprete $p_{x'}$ as an ε-distored image of p_x. Having in mind Th.7 we can construct an ε-partition of the set of the features P (resp. of W). The elements of the ε-class are ε-equivalent (and ε-distored) in comparison with the center p_x. It follows that p_x can be select as an essential feature (sample) for the recognizing problem. But the choise of the essential features is algorithmically decidable (Th.9) because it is equivalent to the constructing of the ε-reduct A_r for A. With the same notions, if A_r is an ε-reduct of A, then A recognizes images, which are ε-distored in comparison with the images acceptable by A_r.

REFERENCES

1. D.Dubois, H.Prade, "Fuzzy Sets and Systems: Theory and Applications", Acad.Press, New York/London, 1980.

2. K.Fu, "Syntactic Methods in Pattern Recognition", Acad. Press, New York/London, 1974.

3. G.Grätzer, "General Lattice Theory", Birkhaüser Verlag, Basel, 1978

4. E.Santos, On reduction of Maxi-min Machines, J.Math.An. Appl., 40, 1972.

5. P.Starke, Abstracte Automaten, Berlin, 1969.

6. V.Topencharov, K.Peeva, Equivalence, Reduction and Minimization of Finite Fuzzy Automata, j.Math.An.Appl. 84, 1981.

Fuzzy automaták és fuzzy grammatikák

K.G. Peeva

Összefoglaló

Legyen A egy véges fuzzy akceptor és R(A) az A által felismerhető szavak halmaza. A szerző bebizonyitja, hogy minden fuzzy szabályos G_F grammatikához, amely generálja az $L(G_F)$ nyelvet, létezik egy A véges fuzzy akceptor ugy hogy $R(A) = L(G_F)$, és vice versa.

Расплывчатые /fuzzy/ грамматики и множества

К.Г. Пеева

Р е з ю м е

Пусть А есть конечный акцептор и R(A) есть множество всех слов, распознанных акцептором А. Доказывается, что для всех регулярных грамматик G_F ненерирующих язык $L(G_F)$, существует конéчный расплывчатый акцептор такой, что $R(A)=L(G_F)$ и наоборот.

MTA SZTAKI Közlemények 33/1985 p. 69-85

О ЛИНЕЙНОЙ СЛОЖНОСТИ РЕАЛИЗАЦИИ НЕКОТОРЫХ ОПЕРАТОРОВ СДВИГА

Р.Л. Шчепанович

Математический институт,
Университет Новы Сад, г. Новы Сад,
Ю г о с л а в и я

Будем рассматривать реализацию одного класса операторов в классе схем из функциональных элементов в базисе &, V, - /определение см., например, в [1]/. Сложность L(S) схемы S определим как число элементов в ней. Остальные функции Шеннона определим как обычно: сложность L(F) оператора F, это наименьшая из сложностей схемы, реализующих F, и наконец, сложность $L(\mathcal{F})$ класса операторов $\mathcal{F}$ - это $\max_{F \in \mathcal{F}} L(F)$.

Пусть B_n множество всех наборов $\tilde{\alpha} = (\alpha_0, \ldots, \alpha_{n-1})$ из нулей и единиц. Отображение B_n в B_m будем называть (n,m) - оператором. Через $|\tilde{\alpha}|$ обозначим $\sum_{i=0}^{n-1} \alpha_i 2^i$ и через $||\tilde{\alpha}||$ - число единиц в наборе $\tilde{\alpha}$.

Обозначим через $T_n(n+\rceil\log(n+1)\lceil, n)$ - оператор, который сдвигает направо произвольный набор $\tilde{x}$ на произвольное число $|\tilde{\tilde{y}}|$. /Символ $\rceil a\lceil$ означает наименьшее целое число, не меньшее a/. Более точно оператор T_n по двум наборам $\tilde{x} = (x_0, \ldots, x_{n-1})$ и $\tilde{y} = (y_0, \ldots, y_{\rceil\log(n+1)\lceil - 1})$ выдает следующий набор $\tilde{z} = (o, \ldots, o, x_0, \ldots, x_{n-|\tilde{y}|-1})$. Обозначим его через $\tilde{x} \to |\tilde{y}|$. Известна следующая оценка О.Б. Лупанова сложности реализации оператора T_n:

Л е м м а 1.[2]

$$L(T_n) \leq C_T \cdot n \cdot \log n$$

Всюду в этой статье буквой C /с индексами, штрихами и т.д./

обозначаются некоторые константы, символ log означает логарифм по основанию 2 и через ℓ_n будем обозначать величину $]\log(n+1)[$.

О.Б. Лупановым была высказана гипотеза, что оператор T_n не допускает линейной реализации.

Интересной является следующая задача: какова сложность реализации класса $\mathcal{T} = \{T^{\tilde{\alpha}} | \tilde{\alpha} = (\alpha_0,\ldots,\alpha_{n-1}) \in B_n\}(\ell_n, n)$ - операторов, которые получаются из оператора T_n подстановками констант $\alpha_0,\ldots,\alpha_{n-1}$ на место переменных $x_0,\ldots,x_{n-1}$?

Здесь доказывается линейность реализации одного класса операторов $T^{\tilde{\alpha}}$, а именно тех, которые определены множеством наборов $\tilde{\alpha}$, у которых, по порядку, не больше $\frac{\log n}{\log\log n}$ единиц.

Введем обозначения: $V_{n,k} = \{\tilde{\alpha} | \tilde{\alpha} \in B_n, \|\tilde{\alpha}\| = k\}$,
$\mathcal{T}_A = \{T^{\tilde{\alpha}} | \tilde{\alpha} \in A \subseteq B_n\}$

Т е о р е м а. Для любого $k \le C \frac{\log n}{\log\log n}$, $L(\mathcal{T}_{V_{n,k}}) \le C_{T_k} \cdot n$.

Сначала введем вспомогательные операторы, а потом сформулируем и докажем несколько лемм, из которых и будет следовать утверждение теоремы.

1^o. Оператор K_m /"дешифратор"/. Это $(m, 2^m)$ - оператор, который преобразует любой набор $\tilde{x} = (x_0,\ldots,x_{m-1})$ в набор $\tilde{y} = (y_0,\ldots,y_{2^m-1})$ такой, что

$$y_i = \begin{cases} 1, & \text{если } i = |\tilde{x}| \\ 0, & \text{если } i \neq |\tilde{x}| \end{cases}$$

Хорошо известен следующий факт.

Л е м м а 2.1. $L(K_m) \le C_K \cdot 2^m$.

2^o. Оператор E_m /в некотором смысле является обратным к K_m/. Это $(2^m, m+1)$ - оператор. Он набор $\tilde{\xi}_i = (o,\ldots,o,\underset{i}{1},o,\ldots,o)$

длины 2^m преобразует в двоичную запись номера разряда, в котором стоит единица, нулевой набор преобразует в набор $(0,\ldots,0,1)$, а на остальных наборах он может быть произвольным.

Л е м м а 2.2.[2] $L(E_m) \le C_E \cdot 2^m$.

3^o. Оператор умножения $U_{m,\ell}$. Это $(m+\ell,\ m+\ell)$ - оператор. Он по наборам $\tilde{x}$ длины m и $\tilde{y}$ длины ℓ находи $m+\ell$ разрядов их произведения

$$U_{m,\ell}(\tilde{x},\tilde{y}) = \tilde{z} \ , \text{ где } \ |\tilde{z}| = |\tilde{x}| \cdot |\tilde{y}| \ .$$

Л е м м а 2.3. $L(U_{m,\ell}) \le C_U \cdot m \cdot \ell$.

Доказательство легко вытекает из обычного "школьного" алгоритма.

4^o. Операторы деления D_m^k определены натуральным числом k, $1 \le k < 2^m$. Для данного k это $(m,\ 2m)$ - оператор, который по набору $\tilde{x}$ длины m выдает наборы: $\tilde{y}$ длины m такой, что $|\tilde{y}| = \left[\frac{|\tilde{x}|}{k}\right]$ и $\tilde{z}$ длины m такой, что $|\tilde{z}| = |\tilde{x}| - |\tilde{y}| \cdot k$. /Символ $[a]$ означает наибольшее целое число - меньше a./

Л е м м а 2.4. Для любого k, $1 \le k < 2^m$,

$$L(D_m^k) \le C_D \cdot 2^m .$$

Доказательство тривиальное.

5^o. Операторы ограниченного сдвига $M^{\tilde{\alpha}}$, $\tilde{\alpha} \in B_n$. Для данного набора $\tilde{\alpha}$, $M^{\tilde{\alpha}}$ это $(\ell_n,\ n)$ - оператор, который произвольный набор $\tilde{x}$ длины ℓ_n преобразует в набор $\tilde{\alpha}^{\rightarrow}_{|\tilde{x}|}$, если $|\tilde{x}| < C_{25} \frac{n}{||\tilde{\alpha}||}$, и в нулевой набор длины n, в остальных случаях.

Л е м м а 2.5. Для любого $\tilde{\alpha} \in B_n$,

$$L(M^{\tilde{\alpha}}) \le C_M \cdot n .$$

Доказательство. Обозначим через p_i, $i = 1,2,\ldots,||\tilde{\alpha}||$ номера разрядов, в которых находятся единицы в наборе $\tilde{\alpha}$. Схема для $M^{\tilde{\alpha}}$ строится в соответствии со следующим алгоритмом /рис. 1/:

1. Входной набор $\tilde{x}$ подается на входы "дешифратора" K_{ℓ_n}. $L(K_{\ell_n}) \le C'_K \cdot n$ /лемма 2.1/.

2. Младших $C_{25}\frac{n}{\|\tilde{\alpha}\|}$ разрядов выходного набора "дешифратора" /или он целиком, если $\|\tilde{\alpha}\| \le C_{25}$/ подаются на "вторые" входы $\|\tilde{\alpha}\|$ - штук схем A_{p_i}, $i = 1,\ldots,\|\tilde{\alpha}\|$, где схема A_{p_i} сдвигает i-ю единицу набора $\tilde{\alpha}$. На "первые" входы схемы A_{p_1} подается нулевой набор длины n, а на "первые" входы схемы A_{p_i}, $i > 1$, подается выходной набор схемы $A_{p_{i-1}}$ /т.е. набор, у которого первых $(i-1)$ единиц уже поставлены на свои места/. Более формально схема A_p, $0 \le p \le n-1$ реализует $(n + C_{25}\frac{n}{\|\tilde{\alpha}\|}, n)$ - оператор, который по наборам $\tilde{\alpha} = (\alpha_0,\ldots,\alpha_{n-1})$ и $\tilde{\beta} = (\beta_0,\ldots,\beta_{C_{25}\frac{n}{\|\tilde{\alpha}\|}-1})$ вычисляет набор $\tilde{\gamma} = (\gamma_0,\ldots,\gamma_{n-1})$ такой, что

$$\gamma_i = \begin{cases} \alpha_i \vee \beta_{i-p}, & p \le i < \min\{n, C_{25}\frac{n}{\|\tilde{\alpha}\|} + p\} \\ \alpha_i, & \text{в остальных случаях.} \end{cases}$$

Очевидно, для любого i, $L(A_{p_i}) \le C_{25}\frac{n}{\|\tilde{\alpha}\|}$.

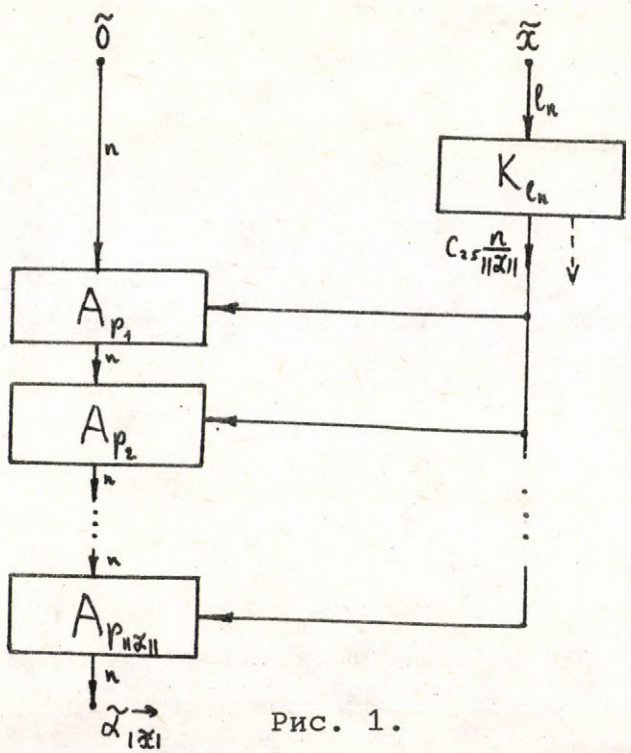


Рис. 1.

Таким образом, $L(M^{\tilde{\alpha}}) \le L(K_{\ell_n}) + \sum_{i=1}^{\|\tilde{\alpha}\|} L(A_{p_i}) \le C_M \cdot n$.

Лемма доказана.

Теперь сформулируем и докажем четыре леммы, из которых и будет следовать утверждение теоремы.

Л е м м а 3. Пусть набор $\tilde{\alpha} \in B_n$ такой, что между любыми двумя соседними единицами в наборе существует хотя бы (K-1) нулевей, где $K \ge C_3' \log n \cdot \log \log n$. Тогда,

$$L(T^{\tilde{\alpha}}) \le C_3 \cdot n \ .$$

Доказательство. Введем обозначения: $K_1 = [\log K]$, $K' = 2^{K_1}$, $\tilde{\gamma} = (\gamma_0, \ldots, \gamma_{\ell_{n-1}})$ - двоичная запись числа $(K_1 + 1)$ и $\tilde{y} = (y_0, \ldots, y_{\ell_{n-1}})$ - набор, которым задается величина сдвига. Схема для оператора $T^{\tilde{\alpha}}$ строится в соответствии со следующим алгоритмов /см. рис. 3/:

1. По набору $\tilde{y}$ определяются наборы $\tilde{\sigma}_1$ и $\tilde{\sigma}_2$ - двоичные записи чисел $[\frac{|\tilde{y}|}{K'}]$ и $|\tilde{y}| - [\frac{|\tilde{y}|}{K'}] \cdot K'$. Это делает схема $D^{K'}_{\ell_n}$. $L(D^{K'}_{\ell_n}) \le C_D' \cdot n$ /лемма 2.4./.

2. Набор $\tilde{\sigma}_2$ подается на входы схемы $M^{\tilde{\alpha}}$, которая сдвигает набор $\tilde{\alpha}$ на число $|\tilde{\sigma}_2| < K' \le K \le C'' \frac{n}{\|\tilde{\alpha}\|} \cdot L(M^{\tilde{\alpha}}) \le C_M \cdot n$ /лемма 2.5./.

3. Наборы $\tilde{\sigma}_2$ и $\tilde{\gamma}$ подаются на входы схемы U_{ℓ_n, ℓ_n}, которая их умножает. $L(U_{\ell_n, \ell_n}) \le C_U' \cdot \log^2 n$ /лемма 2.3./.

4. Выходы схемы $M^{\tilde{\alpha}}$ разобъем слева направо на $]n/K'[$ групп, по K' выходов в каждой /кроме, может быть, последней/ и подаем их на входы $]n/K'[$ - штук "кодеров" E_{K_1} /последними, на оставшиеся свободными входы, подаем нули!/. $L(E_{K_1}) \le C_E \cdot K'$ /лемма 2.2/.

5. Объединяя выходы этих "кодеров", их $]n/K'[\cdot (K_1+1)$ штук,

подаем их вместе с $\ell_{\rceil n/K'\lceil\cdot(K_1+1)}$ левыми выходами схемы U_{ℓ_n,ℓ_n} на входы схемы, которая реализует оператор $T_{\rceil n/K'\lceil\cdot(K_1+1)}$, т.е. сдвигает закодированный набор, полученный на выходах "кодеров" E_{K_1}, на число $\lceil\frac{|y|}{K'}\rceil(K_1+1)$. $L(T_{\rceil n/K'\lceil\cdot(K_1+1)})\le C'_T\cdot\rceil n/K'\lceil\cdot K_1\cdot\log n$ /лемма 1./.

6. Выходы схемы $T_{\rceil n/K'\lceil\cdot(K_1+1)}$ разобъем слева направо на $\rceil n/K'\lceil$ групп, по (K_1+1) штук в каждой. В каждой группе, левых K_1 выходов подаются на входы дешифратора K_{K_1} и полученный набор на его выходах поразрядно умножается на последний выход в группе. Обозначим череб $\tilde{\beta}$ набор длины n, который получается на левых n из $\rceil n/K'\lceil\cdot K'$, таким образом, полученных выходах. $L(K_{K_1})\le C_K\cdot K'$ /лемма 2.1./.

7. Набор $\tilde{y}$ подается на входы дешифратора $K_{\ell_n}\cdot L(K_{\ell_n})\le C'_K\cdot 2^n$. /лемма 2.1./.

8. Выходы этого дешифратора подаются на входы схемы H_n, которая любой набор вида /0,...,0,1,0,...,0/ длины n преобразует в набор /0,...,0,1,1,...,1/ длины n, а на остальных наборах она произвольная /см. рис. 2/. Очевидно, $L(H_n)\le n$.

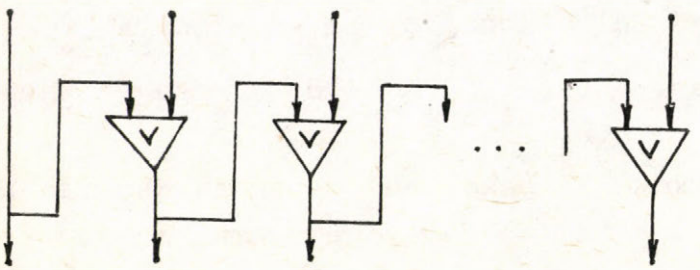


Рис. 2.

9. Набор, реализованный на выходах схемы H_n, поразрядно умножается на набор $\tilde{\beta}$, полученный в шаге 6.

Таким образом, учитывая, что $K \geq C_3' \cdot \log n \cdot \log \log n$, $L(T^{\tilde{\alpha}}) \leq C_3 \cdot n$.

Лемма доказана.

Введем одно понятие. Любой поднабор $(\alpha_i, \alpha_{i+1}, \ldots, \alpha_{i+s})$ набора $\tilde{\alpha} = (\alpha_0, \ldots, \alpha_{n-1})$ называется пустым куском, если все α_j равны 0, $j = i, i+1, \ldots, i+s$. Поднабор, который этому не удовлетворяет, будем называть непустым куском.

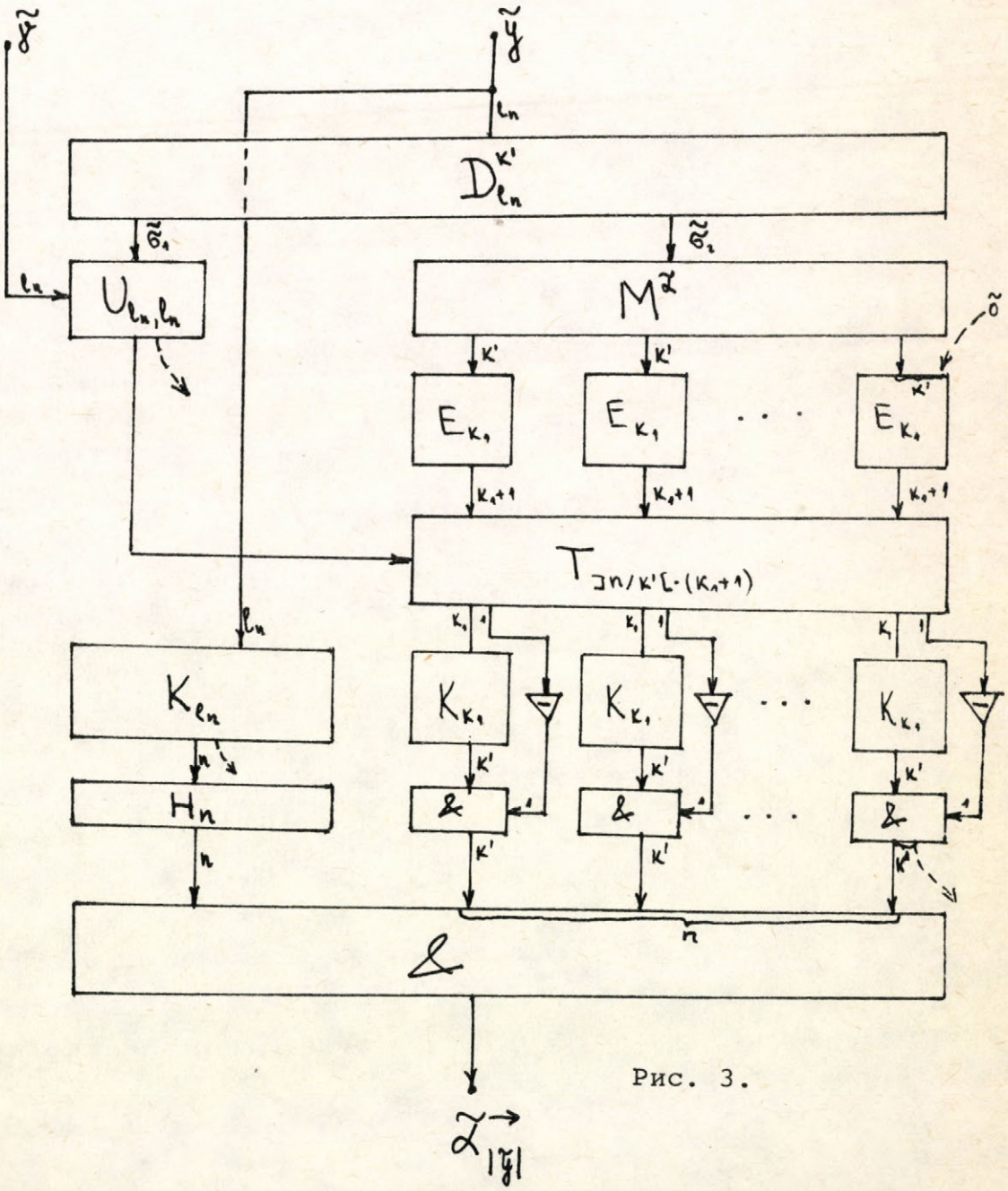


Рис. 3.

Л е м м а 4. Пусть набор $\tilde{\alpha} \in B_n$ такой, что если его рабобъем на куски длины K, $K \le \frac{n}{\log n}$, то его единицы окажутся не более C'_4 кусков. Тогда

$$L(T^{\tilde{\alpha}}) \le C_4 \cdot n.$$

Доказательство. Нетрудно убедиться, что можно предположить, что только один кусок длины K в наборе $\tilde{\alpha}$ непустой. Из этого куска и куска такой же длины, но состоящего только из нулей, образуем набор $\tilde{\sigma}$ длины 2K /см. рис. 4./.

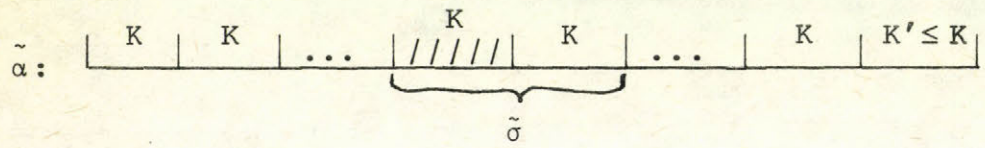


Рис. 4.

Обозначим через $\tilde{y}$ набор, которым задается величина сдвига, через $\tilde{\nu}$ набор длины $]\frac{n}{K}[$, где $\nu_i = 1$ тогда и только тогда, когда i-й кусок набора $\tilde{\alpha}$ непустой. Схема для оператора $T^{\tilde{\alpha}}$ строится в соответствии со следующим алгоритмом /рис. 5./:

1. По набору $\tilde{y}$ определяются наборы $\tilde{\sigma}_1$ и $\tilde{\sigma}_2$ - двоичные записи чисел $[\frac{|\tilde{y}|}{K}]$ и $|\tilde{y}| - [\frac{|\tilde{y}|}{K}]\cdot K$. Это делает схема $D^K_{\ell_n}$. $L(D^K_{\ell_n}) \le C'_D \cdot n$ /лемма 2.4./.

2. Наборы $\tilde{\sigma}$ и младших ℓ_{2K} разрядов набора $\tilde{\sigma}_2$ подаются на входы схемы, которая реализует оператор T_{2K}, т.е. сдвигает набор $\tilde{\sigma}$ на величину $|\tilde{\sigma}_2|$. $L(T_{2K}) \le C_T \cdot 2K \cdot \log(2K)$ /лемма 1./.

3. Младшие $\ell_{]\frac{n}{K}[}$ разряды набора $\tilde{\sigma}_1$ подаются на входы схемы $M^{\tilde{\nu}}$, которая сдвигает набор $\tilde{\nu}$ на величину $|\tilde{\sigma}_1|$. Обозначим этот

сдвинутый набор через $\tilde{\delta} = (\delta_0, \ldots, \delta_{]\frac{n}{K}[-1})$. $L(M^{\tilde{\nu}}) \le C_M \cdot]\frac{n}{K}[$ /лемма 2.5./.

4. С помощью схем операторов B_i, $i = 0,1,\ldots,]\frac{n}{K}[-1$ "поставим" набор, реализованный на выходе схемы T_{2K} на место, которое соответствует величине сдвига /оно определено положением единицы в наборе $\tilde{\delta}$ /. Формально, B_i это $(n+2K+1,n)$ - оператор, который по наборам $\tilde{\gamma}$ длины n, $\tilde{\sigma}'$ длины 2K и разряду δ_i определяет набор $\tilde{z} = (z_0,\ldots,z_{n-1})$ такой, что

$$z_j = \begin{cases} \gamma_j \vee \sigma'_{j-i\cdot K} \,\&\, \delta_i \,, & i\cdot K \le j < \min\{n,(i+2)\cdot K\} \\ \gamma_j \,, & \text{в остальных случаях} \end{cases}$$

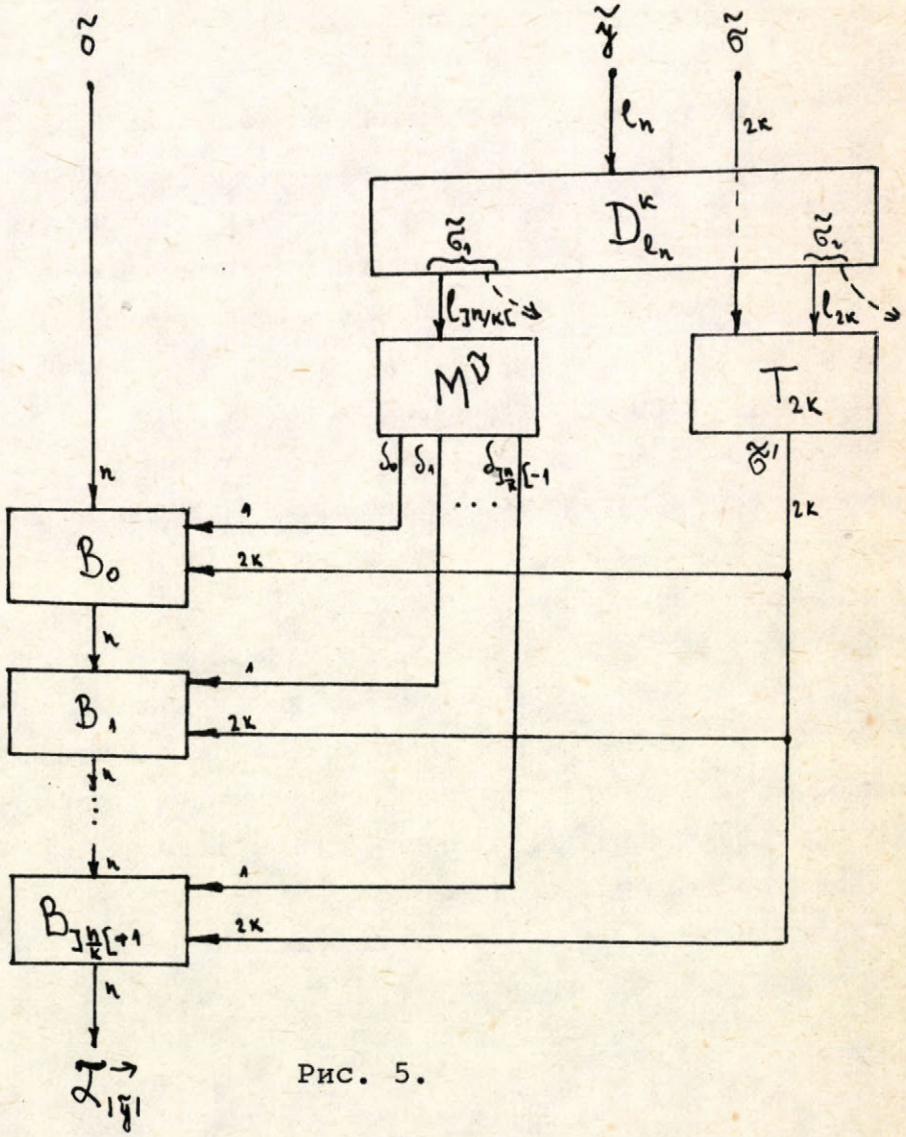


Рис. 5.

Очевидно, что для любого i, $L(B_i) \le C_B \cdot K$. Таким образом,

$$L(T^{\tilde{\alpha}}) \le C'_D \cdot n + C_T \cdot 2K \cdot \log(2K) + C_M \left\lceil \frac{n}{K} \right\rceil + \left\rceil \frac{n}{K} \right\lceil \cdot C_B \cdot K \le C_4 \cdot n \ .$$

Лемма доказана.

Л е м м а 5. Пусть набор $\alpha \in B_n$ можно разбить на куски длины К /кроме последнего/, $K \ge C'_5 \cdot \log n \cdot \log \log n$, так чтобы любыми двумя соседними непустыми кусками, пустых кусков было больше, чем единиц в правом из этих двух непустых кусков. Тогда

$$L(T^{\tilde{\alpha}}) \le C_5 \cdot n \ .$$

Доказательство. Пусть набор $\tilde{\alpha}$ удовлетворяет данному условию. Будем преобразовывать набор $\tilde{\alpha}$, сдвигая его единицы так, что получим набор $\tilde{\beta}$, который удовлетворяет условию леммы 3.

Рассмотрим произвольный непустой кусок в наборе $\tilde{\alpha}$. Обозначим через ℓ число единиц в нем. По условию леммы, левее от этого куска существует $\ell+1$ пустых кусков. Единицы из непустого куска разместим по пустым кускам следующим спомобом: последнюю /если смотреть справа налево/ сдвинем влево на К разрядов, предпоследнюю - на 2К разрядов и т.д., ℓ-ю на $\ell \cdot K$ разрядов. Если это сделаем с любым непустым куском, получим набор $\tilde{\beta}$. Разобъем его на куски длины К. Он обладает следующими свойствами: в каждом куске нет больше одной единицы; расстояние между соседними единицами >К; между соседними единицами, которые принадлежали одному куску в наборе $\tilde{\alpha}$, нет пустых кусков, а между соседними единицами, которые принадлежали разным кускам в наборе $\tilde{\alpha}$ существуют хотя бы два пустых куска. Схему будем строить так, что сначала будем сдвигать набор $\tilde{\beta}$, а потом инверсно описанному выше алгоритму, восстанавливать сдвинутый набор $\tilde{\alpha}$. Ради однозначности декодировки сдвинутого набора "удлиним" набор $\tilde{\beta}$ направо на нулевой кусок длины $(\left\rceil \frac{n}{K} \right\lceil + 1) \cdot K - n$. Полученный набор обозначим через $\tilde{\beta}'$. Теперь опишем алгоритм, в соответствии с которым будем строить схему для оператора $T^{\tilde{\alpha}}$ /см. рис. 8./:

1. Набор $\tilde{y}$, которым задается величина сдвига, подается на выходы схемы, которая реализует оператор $T^{\tilde{\beta}'}$. $L(T^{\tilde{\beta}'}) \le C_3 \cdot n$ /лемма 3./.

2. Выходы этой схемы разобъем на $]\frac{n}{K}[+1$ зон, по К - штук в каждой. Обозначим через $\tilde{\nu}_i$ набор, который реализуется в i-ой зоне, $i = 1,2,\ldots,]\frac{n}{K}[+1$. Наборы $\tilde{\nu}_i$, $i = 1,\ldots,]\frac{n}{K}[+1$ подаются на столько же дизъюнктивных схем V_i, которые обнаруживают присутствие единицы в i-ой зоне, т.е. вычисляются сигналы $t_i = \bigvee_{j=0}^{K-1} \nu_{ij}$. Очевидно, для любого i, $L(V_i) \le K$.

3. По сигналам t_j, $]\frac{n}{K}[$ штук схем R_i вычисляют сигналы r_i, где $r_i = 0$ в том случае, когда единицу из i-ой зоны надо отнести в левый собирающий кусок /см. шаг 5. и рис. 5./, и $r_i = 1$, если эту единиуц надо отнести в правый собирающий кусок /в случае, когда непустой кусок набора $\tilde{\alpha}$ при сдвиге частично попал в два куска длины К/. Нетрудно убедиться, что должно быть: $r_1 = 0$ и $r_{i+1} = \bar{r}_i \cdot t_i \cdot \bar{t}_{i+1} \cdot t_{i+2} \vee r_i t_{i+1}$, $i = 1,2,\ldots,]\frac{n}{K}[-1$. Таким образом, $L(R_i) \le C_R$ для любого i.

4. По сигналам t_s и r_ℓ, $]\frac{n}{K}[$ штук схем P_i, $i = 1,\ldots,]\frac{n}{K}[$ вычисляют сигналы p_i, где $p_i = 1$, когда в i-й и (i+1)-й кусок надо поставить левый и правый собирающий кусок /потому, что единицы "вернулись" на свое место/ и $p_i = 0$ в остальных случаях. Нетрудно убедиться, что должно быть $p_i = \bar{t}_i \cdot \bar{t}_{i+1} \vee t_i \cdot \bar{t}_{i+1} \cdot r_i$, $i = 1,\ldots,]\frac{n}{K}[$ откуда и следует оценка $L(P_i) \le C_p$ для любого i.

5. Для каждого i, $i = 1,\ldots,]\frac{n}{K}[$ строится схема W_i, которая определяет набор $\tilde{\sigma}_i$ длины 2К, т.е. левый и правый собирающий кусок. Более подробно, схема W_i зависимо от значения сигнала r_i, пропускает единицу из i-ой зоны /набора $\tilde{\nu}^i$/ в левый или правый кусок набора $\tilde{\sigma}_{i-1}$, умноженного сначала поразрядно на p_{i-1} /см. рис. 6./. Для $\tilde{\sigma}_0$ берется нулевой набор длины 2К, $p_0 = 1$.

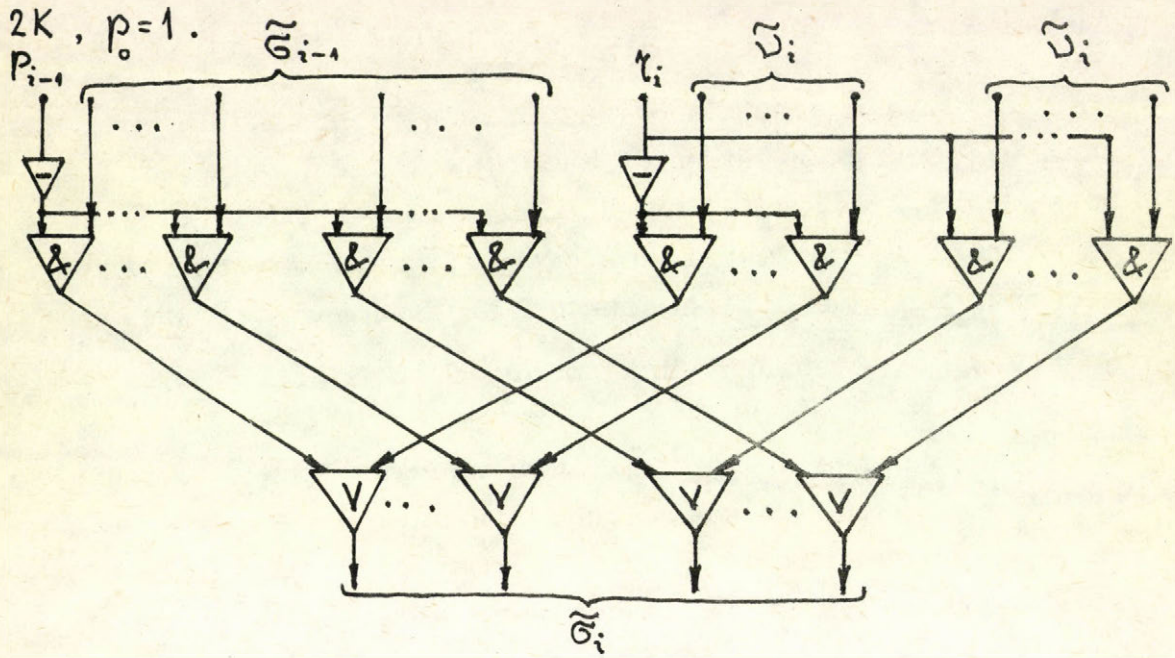


Рис. 6.

Очевидно, что $L(W_i) \le C_W \cdot K$ для любого i.

6. Наборы $\tilde{\sigma}_i$, $i = 1,\dots,\left]\frac{n}{K}\right[$ умножаются поразрядно на p_i при помощи $\left]\frac{n}{K}\right[$ штук схем G_i. Очевидно, что $L(G_i) \le C_G \cdot K$ для любого i.

7. Обозначим наборы, которые получаются на выходах схемы G_i через $\tilde{\sigma}'_i$, $i = 1,\dots,\left]\frac{n}{K}\right[$. При помощи $\left]\frac{n}{K}\right[-1$ штук схем Σ_i будем поразрядно суммировать правый кусок набора $\tilde{\sigma}'_i$ с левым куском набора $\tilde{\sigma}'_{i+1}$ /см. рис. 7./.

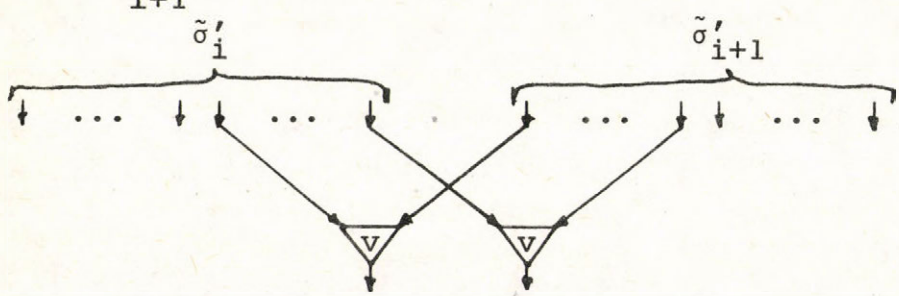


Рис. 7.

Очевидно, что $L(\Sigma_i) \le C_\Sigma \cdot K$ для любого i. Объединяя оставшиеся выходы схемы G_1 с выходами схемы Σ_i, $i = 1, \ldots,]\frac{n}{K}[-2$ и $n - (]\frac{n}{K}[-1)\cdot K$ выходов схемы $\Sigma_{]\frac{n}{K}[-1}$, получим на этих выходах набор $\tilde{\alpha}_{|\tilde{y}|}^{\rightarrow}$.

Таким образом, $L(T^{\tilde{\alpha}}) \le C_6 \cdot n$.

Лемма доказана.

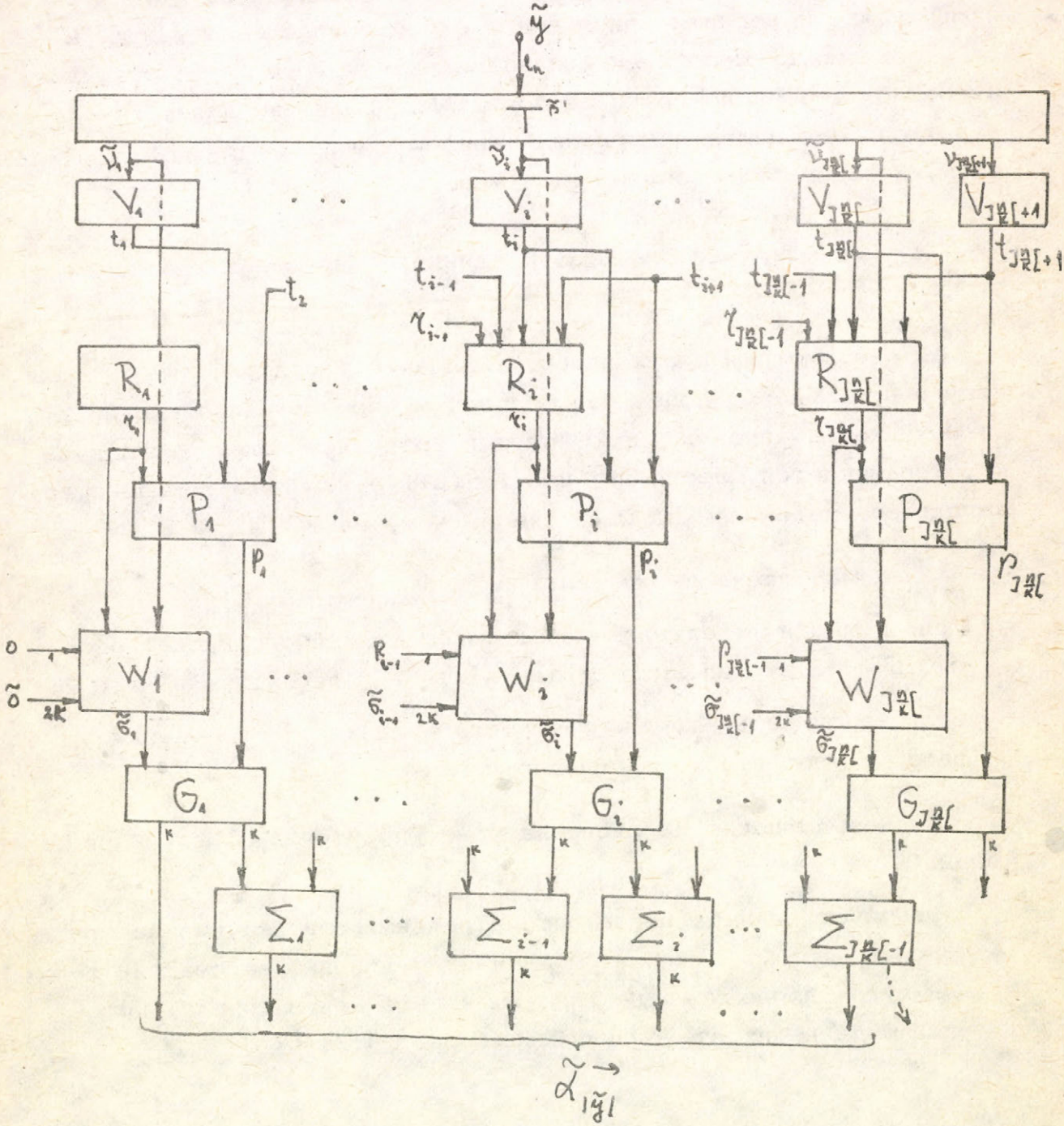


Рис. 8.

Л е м м а 6. Любой набор $\tilde{\alpha} \in B_n$, $\|\tilde{\alpha}\| = K$, можно разложить в поразрядную сумму по mod2 двух наборов: один удовлетворяет условию леммы 5, а у второго число разрядов между первой и последней единицами $< K^K \cdot M$, где $M = \log n \cdot \log \log n$.

Доказательство. Разобъем единицы в наборе $\tilde{\alpha}$ на группы так, чтобы между соседними единицами в одной группе было меньше M разрядов, между соседними единицами разных групп - $\geq M$ разрядов. Очевидно, что число групп не больше K. Обозначим длину /число разрядов/ самой длинной группы через D_1. Теперь приведем алгоритм, которым будем объединять "меньшие" группы в "большие".

1. Рассмотрим самую правую единицу в наборе $\tilde{\alpha}$ и положим $i = 1$.

2. Рассмотрим кусок набора $\tilde{\alpha}$ длины D_i, содержащий эту единицу и D_{i-1} разряд левее ее. Обозначим через p число единиц в этом куске, а через r число "пустых" разрядов левее этого куска. Спрашивается, верно ли, что $r \geq (p+1)D_i$? Если неравенство выполнено, переходим на шаг 3, если нет, на шаг 4.

3. У нас две возможности:

а/ если существуют единицы левее рассматриваемого куска, рассмотрим первую из них /справа налево/ и переходим на шаг 2;

б/ если не существуют, алгоритм останавливается и, очевидно, набор $\tilde{\alpha}$ удовлетворяет условию леммы 5.

4. Невыполнение неравенства может произойти по двум причинам:

а/ существует единица левее рассматриваемого куска, из-за которой $r < (p+1)D_i$. В этом случае, "удлиним" рассматриваемый кусок влево пока не включим в него целую группу, которой принадлежит эта единица /см. рис. 9./.

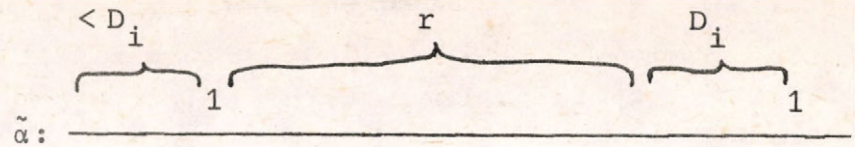


Рис. 9.

Длину полученного таким образом куска обозначим через D_{i+1}. Очевидно, $D_{i+1} < D_i + r + D_i < (p+3)D_i$. Опять возвращаемся к самой правой единице в наборе $\tilde{\alpha}$, но уже с новым D_i /i на единицу больше/ и переходим на шаг 2.

б/ Не существует единиц левее рассматриваемого куска. Алгоритм останавливается.

Таким образом, алгоритм останавливается либо в шаге 3 б/, и набор $\tilde{\alpha}$ удовлетворяет условию леммы 5., либо в шаге 4 б/, когда набор $\tilde{\alpha}$ можно поразрядно разложить на сумму по mod2 набора, который удовлетворяет условию леммы 5. /это правый кусок набора или нулевой набор/ и набора, у которого между крайними единицами $< D_j$ разрядов, для некоторого j. Нетрудно заметить, что при каждом выполнении шага 4 а/ алгоритма, число групп в наборе $\tilde{\alpha}$ уменьшается хотя бы на единицу. Следовательно $j < K$. Из $D_{i+1} < (p+3)D_i$ и $D_1 < K \cdot M$ получим $D_j < K^K \cdot M$ для любого j.

Лемма доказана.

Доказательство теоремы непосредственно вытекает из лемм 4., 5. и 6. и факта, что при $K \le \frac{\log n}{\log \log n}$ выполнено неравенство $K^K \cdot M < C' \frac{n}{\log n}$.

Л И Т Е Р А Т У Р А

[1] Лупанов О.Б.: О синтезе некоторых классов управляющих систем. В сб.: Проблемы кибернетики, М., 1963, вып. 10, стр. 63-97.

[2] Лупанов О.Б.: Об одном подходе к синтезу управляющих систем - принципе локального кодирования. - В сб.: Проблемы кибернетики, М., 1965, вып. 14, стр. 31-110.

Bizonyos eltolás operátorok realizációinak lineáris komplexitása

R.L. Ščepanovič

Összefoglaló

Jelöljük $L(F)$-el az F operátor komplexitását /l. [1]/, és $L(\mathcal{F}) = \max_{F\in\mathcal{F}} L(F)$ jelentse az $\mathcal{F}$ operátorcsalád komplexitását. Lupanov [1] belátta, hogy bizonyos fajta T_n eltolás operátorra $L(T_n) \leq C\, n\cdot\log n$ és azt sejtette, hogy a T_n operátornak nincs olyan realizációja, amelynek komplexitása lineáris volna. A cikkben a szerző bemutat egy olyan $\mathcal{T}$ operátor családot, amely T_n operátoroknak bizonyos módositásait tartalmazza és amelyre $L(\mathcal{T}) \leq c\, n$.

Linear complexity of some translation operators

R.L. Ščepanovič

Summary

Denote by $L(F)$ and $L(\mathcal{F}) = \max_{F\in\mathcal{F}} L(F)$ the complexity of the operator F and the operator family $\mathcal{F}$ respectively. (see [1]). Lupanov [1] proved that $L(T_n) \leq c\, n\cdot\log n$ for some translation operators T_n and conjectured that T_n has no linear realization. In the paper a family $\mathcal{T}$ of modified operators T_n is shown, such that $L(\mathcal{T}) \leq c\, n$.

MTA SZTAKI Közlemények 33/1985 p. 87-94

COMMENTS ON A PROBLEM OF CONTINUOUS AND PERIODIC FUNCTIONS

L. Zs. VARGA

BME Folyamatszabályozási Tanszék
Budapest

Let us consider the following problem:

Problem: Let $f(x)$ be continuous and of period one on the real line. If d_j, $j = 1,2,\ldots,n$, are n numbers such that each $d_j - d_1$ is irrational, then there are two rational numbers r and r' for which

$$(1) \quad f(r) \leq f(r+d_j) \text{ and } f(r') \geq f(r'+d_j), \quad j=1,2,\ldots,n.$$

J.S. Hwang has proved [1] the problem but there are some mistakes in the proof. In this paper a correct proof for the Problem is given.

1. A lemma. In order to prove Problem will use the following lemma:

Lemma 1: Let d_1 be irrational and let $d_j - d_1$ be rational, $j = 2,3,\ldots,n$. If $I_j(k)$, $j = 1,2,\ldots,n$, are non-negative integral valued functions such that

$$\sum_{j=1}^{n} I_j(k) = k, \quad \text{for each} \quad k = 1,2,\ldots,$$

then there are a and b real numbers such that the points (a_k) are dense in (a,b), where

$$a_k = \sum_{j=1}^{n} I_j(k) d_j, \quad \text{for} \quad k = 1,2,\ldots$$

and $0 \leq a < b \leq 1$.

Proof: We know that $d_j = d_1 + p_j/q_j$ for some integers p_j and q_j. If we denote $Q = \prod_{i=1}^{n} q_j$, then we can write:

$$Q_k = \sum_{j=1}^{n} I_j(k) d_j = k \cdot d_1 + \sum_{j=1}^{n} I_j(k) p_j/q_j =$$

$$(4) \qquad = k \cdot d_1 + \frac{\sum_{j=1}^{n} I_j(k) p_j Q/q_j}{Q}$$

$$Q_k = k \cdot d_1 + \ell + m/Q$$

where ℓ, m are integers and $0 \leq m \leq Q$. So from (4) we know that (Q_k) is in one of the following points:
$(k \cdot d_1)$ or $(kd_1 + 1/Q)$ or ... $(kd_1 + (Q-1)/Q)$.

Our proof will be an indirect one. The lemma says that there are numbers a and b such that for each real numbers c_{11} and c_{12} satisfying $a \leq c_{11} < c_{12} \leq b$ we can find k such that $c_{11} < a_k < c_{12}$.

Let us suppose that it is not true. So for each a and b satisfying $0 \leq a \leq b \leq 1$ we can find c_{11} and c_{12} satisfying $a \leq c_{11} < c_{12} \leq b$, for which there is no k such that $c_{11} < (a_k)\ c_{12}$.

Let us take any of the numbers a, b and to them a pair of c_{11} and c_{12} satisfying the above assertion.

If $c_{11} + 1/Q < c_{12} + 1/Q < 1$ then let $c'_{11} = c_{11} + 1/Q$ and

$$c'_{12} = c_{12} + 1/Q.$$

If $c_{11}+1/Q<1\leq c_{12}+1/Q$ then let $c'_{11}=c_{11}+1/Q$ and $c'_{12}=1$.

If $1\leq c_{11}+1/Q<c_{12}+1/Q$ then let $c'_{11}=(c_{11}+1/Q)$ and

$c'_{12}=(c_{12}+1/Q)$.

According to our assumption the points (a_k) are not dense in (c'_{11},c'_{12}), so there are c_{21} and c_{22} satisfying $c'_{11}\leq c_{21}<c_{22}\leq c'_{12}$, for which there is no k such that $c_{21}<(a_k)<c_{22}$.

Moreover there is no k such that $(c_{21}-1/Q)<(a_k)<(c_{22}-1/Q)$ because $c_{11}\leq(c_{21}-1/Q)<(c_{22}-1/Q)\leq c_{12}$.

If $c_{21}+1/Q<c_{22}+1/Q<1$ then let $c'_{21}=c_{21}+1/Q$ and

$c'_{22}=c_{22}+1/Q$.

If $c_{21}+1/Q<1\leq c_{22}+1/Q$ then let $c'_{21}=c_{21}+1/Q$ and

$c'_{22}=1$.

If $1\leq c_{21}+1/Q<c_{22}+1/Q$ then let $c'_{21}=(c_{21}+1/Q)$ and

$c'_{22}=(c_{22}+1/Q)$.

According to our assumption the points (a_k) are not dense in (c'_{21}, c'_{22}), so there are c_{31} and c_{32} satisfying $c'_{21}\leq c_{31}<c_{32}\leq c'_{22}$, for which there is no k such that $c_{31}<(a_k)<c_{32}$.

Moreover there is no k for which either of the following inequalities would be true:

$(c_{31}-1/Q)<(a_k)<(c_{32}-1/Q)$, because

$$c_{21}\leq(c_{31}-1/Q)<(c_{32}-1/Q)\leq c_{22}$$

$(c_{31}-2/Q)<(a_k)<(c_{32}-2/Q)$, because

$$c_{11}\leq(c_{31}-2/Q)<(c_{32}-2/Q)\leq c_{12}\ .$$

Continuing this procedure until we reach $Q-1$ we get, that there is no k for which any of the following inequalities would be true:

$$c_{Q1}<(a_k)<c_{Q2}.$$

$$(c_{Q1}-1/Q)<(a_k)<(c_{Q2}-1/Q) \tag{5}$$

$$(c_{Q1}-2/Q)<(a_k)<(c_{Q2}-2/Q)$$

$$\vdots$$

$$(c_{Q1}-\tfrac{Q-1}{Q})<(a_k)<(c_{Q2}-\tfrac{Q-1}{Q})$$

But the Theorem 445 in [2] gives ghat $(k\ d_1)$ is dense in $(0,1)$, so there is k_o such that

$$(c_{Q1}-\tfrac{Q-1}{Q})<(k_o d_1)<(c_{Q2}-\tfrac{Q-1}{Q}).$$

and we know

$$(c_{Q1}-\tfrac{Q-2}{Q})<(k_o d_1+\tfrac{1}{Q})<(c_{Q2}-\tfrac{Q-2}{Q}) \tag{6}$$

$$\vdots$$

$$(c_{Q1}-1/Q)<(k_o d_1+\tfrac{Q-2}{Q})<(c_{Q2}-1/Q)$$

$$c_{Q1}<(k_o d_1+\tfrac{Q-1}{Q})<c_{Q2}\ .$$

At the beginning of the proof of this lemma we saw that (a_{ko}) is in one of the points $(k_o d_1)$, $(k_o d_1+1/Q),\ldots,$ $(k_o d_1+(Q-1)/Q)$. Considering the inequalities of (6) this contradicts, that there isno k satisfying the inequalities (5). Supposing that our lemma is not true we got to contradiction, so the assertion of our lemma is true.

2. *The proof of Problem*. The proof of Problem is similar to the proof of Theorem 1 in [1], but in the proof we have to use our correct Lemma 1 and to argue differently.

In order to prove the first set of inequalities in (1) let m denote the minimum of $f(x)$, where $f(x)$ is a continuous function of period one and let S_m be

$$S_m = \{x : f(x) = m, \quad 0 \leq x < 1\} \tag{7}$$

If d_1 is irrational and there is apoint $y \in S_m$ such that

$$f(y) < f(y+d_j) \quad \text{for all} \quad j = 1,2,\ldots,n$$

then the proof is the same as in [1].
If d_1 is irrational and for each $x \in S_m$ there is a $j = j(x)$ such that

$$f(x) = f(x+d_j) \qquad 1 \leq j \leq n \tag{8}$$

then applying (8) succesively we obtain a sequence of points in S_m:

$$x,\ x+d_{j1},\ x+d_{j1}+d_{j2},\ldots$$

Which can be represented, using the notations of Lemma 1, as

$$x_k = x + \sum_{j=1}^{n} I_j(k) d_j \qquad k = 1,2,\ldots$$

By our Lemma 1 the points (x_k) are dense in (a,b), where $0 \leq a \leq b \leq 1$. From the contuniuity of $f(x)$ it follows that $f(x) = m$ for each $x \in (a,b)$. In each interval you can find a rational number, so there is an r_o rational number in (a,b). Since $f(r_o) = m$, the first set of inequalities of (1) are true for r_o.

If d_1 is rational then the proof is the same as in the proof of Theorem 1 of [1], but we have to make clear that in the second part of the proof of Theorem 1 of Hwang in [1] S_x means the set of all points (x_k) in $(0,1)$, where

$$x_k = 0 + \sum_{j=1}^{n} I_j(k) d_j$$

and it is not a subset of S_m, only in special cases.

The second set of inequalities in (1) can be obtained similarly by replacing minimum by maximum.

REFERENCES

[1] J.S. Hwang: A Problem on Continuous and Periodic Functions. Pacific Journal of Mathematics Vol.117. No.1. 1985.

[2] G.H. Hardy and E.M. Wright: The Theory of Numbers, Oxford University Press 1945. (Theorem 445)

Megjegyzések a folytonos és periódikus függvények egy problémájához

L.Zs. Varga

Összefoglaló

Tekintsük a következő feladatot:

Legyen $f(x)$ valós számokon értelmezett egységnyi periódusu valós értékü folytonos függvény. Ha adott d_j, $j = 1,2,\ldots,n$; n darab szám ugy, hogy minden j-re $d_j - d_1$ irracionális, akkor van két racionális szám r és r' amelyekre

$$f(r) \leq f(r+d_j) \quad \text{és} \quad f(r') \geq f(r'+d_j); \; j=1,2,\ldots,n.$$

J.S. Hwang bebizonyitotta [1] ezt az állitást, de a bizonyitásban hibákat követett el. Ebben a dolgozatban megadjuk az állitás helyes bizonyitását.

ПРИМЕЧАНИЯ К ПРОБЛЕМЕ НЕПРЕРЫВНЫХ ПЕРИОДИЧЕСКИХ ФУНКЦИЙ

Л.Ж. Варга

Р е з ю м е

Посмотрим следующую задачу:

Пусть $f(x)$ есть непрерывная действительная периодическая функция над вещественными числами с периодом один. Если даны числа d_j, $j = 1,2,\ldots,n$, где такие, что каждая разность $d_j - d_1$ рациональная, тогда существуют два рациональных числа r и r' такие, что

(1) $f(r) \le f(r+d_j)$ и $f(r') \ge f(r'+d_j)$, $j = 1,2,\ldots,n$.

Й.Ш. Хванг доказал утверждение этой задачи [1], но в доказательстве имеются некоторые недостатки. В этой статье мы даем правильное доказательство утверждения этои задачи.

MTA SZTAKI Közlemények 33/1985 p. 95-112

NETS, POLYCATEGORIES AND SEMANTICS OF PARALLEL PROGRAMS

Y.P. Velinov

Center of Applied Mathematics; Higher Institute of Mechanical and Electrical Engineering; 1000 Sofia, p.box 384, Bulgaria

The purpose of the present paper is to establish some connections between (a modified variant of) Petry nets and Polycategories and on this base to spread the category approach to the semantics of programs (flow diagrams) proposed by ADJ group [JCS] and Goguen [HCT] over a wider class of parallel programs.

The reader which is not accustomed with the category approach to programming is invited to consult [HCT] or [JCS], where the ideas not concerned with the parallelism can be easily followed.

1. POLYGRAHS AND λ-POLYCATEGORIES

1.1. A *polygraph* (or a net) $\mathbf{G}$ is defined to be a system presented by the following string of data

$$< \mathbf{O}, \mathbf{A}, +, \bullet, \mathrm{dom}, \mathrm{cod}>$$

where

- $\mathbf{O} \doteq \mathrm{Ob}(\mathbf{G})$ is a set of elements called *objects*; $\mathbf{A} \doteq \mathrm{Ar}(\mathbf{G})$ is a set of elements called *(poly)arrows*; $\bullet \in \mathbf{O}$;
- dom,cod and + are three mappings: $\mathrm{dom}\ \mathbf{A} \to \mathbf{O}$ puts in correspondence to each arrow an object called its *initial object*; $\mathrm{cod}: \mathbf{A} \to \mathbf{O}$ puts in correspondence to each arrow an object called its *final object*; $+: \mathbf{O} \times \mathbf{O} \to \mathbf{O}$.

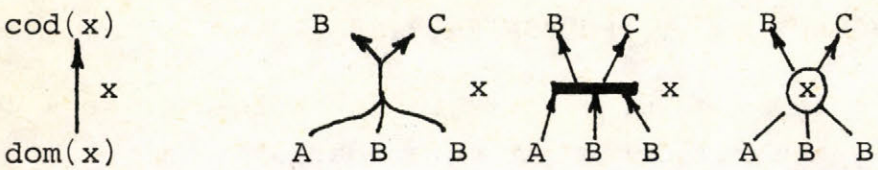
These data are such that:

PO.$<\mathcal{O},+,\bullet>$ is a monoid with operation + and neutral element $\bullet$.

The objects of a polygraph usually will be denoted by A, B,C,u,v,w and the arrows by x,y,z,f,g,h.

The presented structure is similar to the Petry net structure [PNT]. If the monoid of objects of a polygraph is taken to be a free monoid over some set $\mathbb{V}$ of "places" and arrows are taken to be "transitions" the only significant difference will be that instead of sets string of places are put in correspondence to the transitions by the "input" and "output" functions dom and cod.

Situations in a polygraph can be graphically illustrated representing objects by their names situated on a sheet of paper and arrows by "drawn branching arrows" (sometimes with boxes or circules on them) which lead from names of objects to names of objects. If several names of objects are situated at different beginnings (ends) of a drawn arrow, they form the name of the initial (final) object of the arrow by a standard agreement of composing them from the left to the right.



dom(x) = A + B + B, cod(x) = B + C

An ordered triple of objects will be called a *connector*.

Connector will be denoted by $\lfloor$ or $\top$. A connector <A,B,C> will be denoted by $A\underline{B}C$ also. If $\lfloor$ =<A,B,C> is a connector then $|\lfloor|$ will denote the object A+B+C and $L(\lfloor)$, $C(\lfloor)$,$R(\lfloor)$ will denote its first, second and third component respectively. $C(\lfloor)$ will be called the *center* of the connector. The set of all connectors of a polygraph $\mathcal{G}$ will be denoted by Con($\mathcal{G}$).

An operation $\blacklozenge$: Con($\mathcal{G}$)xCon($\mathcal{G}$) $\rightarrow$ Con($\mathcal{G}$) defined by the correspondence

$$\langle\langle A_1, A_2, A_3\rangle, \langle B_1, B_2, B_3\rangle\rangle \mapsto \langle A_1+B_1,\ B_2,\ B_3+A_3\rangle$$

will be used in the following exposition.

The sign "+" for the monoidal operation will often be omitted.

1.2 A λ *-polycategory* $\mathcal{P}$ is defined to be a system presented by the string of data

$$\langle \mathcal{O},\ \mathcal{A}\ +,\ \bullet, \mathrm{dom},\ \mathrm{cod},\ \gamma,\ I\rangle$$

where $\langle \mathcal{O}, \mathcal{A}, +, \bullet, \mathrm{dom}, \mathrm{cod}\rangle \doteq \mathrm{Gr}(\mathcal{P})$ is a polygraph (the underlying polygraph of the polycategory), $I: \mathcal{O} \to \mathcal{A}$, $I{:}A \mapsto I_A$ is a mapping that puts in correspondence to each object a selected arrow, called *identity* arrow and

$$\gamma: \mathcal{A}\mathrm{xCon}(\mathrm{Gr}(\mathcal{P}))\mathrm{x}\,\mathcal{A} \to \mathcal{A}$$
$$\gamma{:}\langle x, \perp, y\rangle \mapsto x \lceil\perp\rceil y$$

is a partial mapping, called *composition law for the arrows*. These data fulfil the following axioms:

PY 1. (Existence of composites)

$$\exists z(x \lceil\perp\rceil y = z) \iff \mathrm{dom}(x) = |\perp| \ \&\ C(\perp) = \mathrm{cod}(y).$$

PY 2. (dom and cod of a composite)

$$x \lceil\perp\rceil y = z \iff \begin{cases} \mathrm{dom}(z) = |\perp \blacklozenge \mathrm{dom}(y)| \\ \mathrm{cod}(z) = \mathrm{cod}(x) \end{cases}$$

PY 3. (Partial commutativity)

$$(x/\overline{A_1\underline{A}_2A_3A_4A_5}/y)/\overline{A_1\mathrm{dom}(y)A_3\underline{A}_4A_5}/z =$$

$$= (x/\overline{A_1A_2A_3\underline{A}_4A_5}/z)/\overline{A_1\underline{A}_2A_3\mathrm{dom}(z)A_5}/y$$

if the described composites exist.

PY 4. (Associativity)

$$x/\overline{\ \perp\ }/(y/\overline{\ \top\ }/z) = (x/\overline{\ \perp\ }/y)/\overline{\perp\bullet\top}/z$$

if the described composites exist.

PY 5. (Unit law)

For any object A and arrows $x,y : \mathrm{dom}(I_A) = \mathrm{cod}(I_A) = A$ and $I_A/\underline{A}/x=x$, $y/\underline{A}/I_A=y$ if the described composites exist.

Now to shorten the notation we can introduce two derivative operations - "o" and "+":

$$x \circ y = x/\overline{\mathrm{dom}(x)}/y,$$
$$x + y = (I_{\mathrm{cod}(x)+\mathrm{cod}(y)}/\overline{\mathrm{cod}(x)\mathrm{cod}(y)}/x)/\overline{\mathrm{dom}(x)\underline{\mathrm{cod}(y)}}/y$$

The main interpretation of the presented structure, which we will need here is the λ-polycategory $\mathbb{P}$fn. It is defined as follows:

- Ob($\mathbb{P}$fn) contains all the sets (in some universe) distinguished to within the associativity of Cartesian product and the product with the set $\{\phi\}$;
- Ar($\mathbb{P}$fn) contains all the functions;
- the monoidal operation is the product and the neutral element of this operation is $\{\phi\}$;
- the composition law for the arrows is the superposition specified by the connectors.

A λ-polycategory is said to be a λ-polycategory with forks iff for each object A there is an arrow $\nabla_A : A \to A+A$ such that

$$(I_{AAA} \overline{\underline{AAA}} \nabla_A) \overline{\underline{AA}} \nabla_A = (I_{AAA} \overline{\underline{AAA}} \nabla_A) \overline{\underline{AA}} \nabla_A \doteq \nabla_A^3$$

More details about polygraphs, polycategories and λ-polycategories can be found in [P,CFP,P.EP,PSMC I, PSMC II, PR,CAP]. In particular λ-polycategories can be viewed as strict monoidal categories [CWM,MFC] in the frame of which a new composition law

$$\langle x, \langle A,B,C\rangle, y\rangle \mapsto x \overline{ABC} y \doteq x \circ (I_A + y + I_C)$$

is introduced [PSMC I, PSMC II, PR].

1.3. Let $\mathcal{G} = \langle \mathbb{V}^*, \mathbb{A}, +, \bullet, dom, cod\rangle$ be a polygraph with a free monoid of objects $\langle \mathbb{V}^*, +, \bullet\rangle$ over a set of nodes $\mathbb{V}$. Notice that in such a situation the objects can be viewed as words over $\mathbb{V}$ and a connector $\perp$ specifies a participation of the word $C(\perp)$ in $|\perp|$.

A sequence of arrow and connectors in $\mathcal{G}$ of the form

$$p = \langle \perp_1, x_1, \perp_2, x_2, \ldots \perp_k, x_k, \ldots, \perp_m, x_m, \ldots \perp_n, x_n, \perp_{n+1}\rangle$$

is called a path in $\mathcal{G}$ iff

- the center of $\perp_i$ is $cod(x_i)$, $(i=1,2,\ldots,n)$;
- $|\perp_{i+1}| = |\perp_i \blacklozenge dom(x_i)|$, $(i=1,2,\ldots,n)$;
- $\perp_{n+1} = \langle \bullet, |\perp_n \blacklozenge dom(x_n)|, \bullet\rangle$.

The objects $|\perp_1|$ and $|\perp_{n+1}|$ are called the *end* and the *beginning* of the path and denoted by $cod^*(p)$ and $dom^*(p)$ respectively.

In degenerated cases, when a path does not contain arrows, it should be in the form $\langle \underline{uu \ldots u}, \underline{u} \rangle$, $u \in V^*$. Such paths are called fork paths (or diagonals). The paths of the form $\langle \underline{u}, \underline{u} \rangle$ are called identity paths. They will be denoted by p_u also. Paths of the form $p_x = \langle \underline{cod(x)}, x, \underline{dom(x)} \rangle$ and degenerated paths will be called *elementary paths*.

Each path p in $\mathcal{G}$ proposes a sequence of words

$$|\perp_1| , |\perp_2| , \ldots , |\perp_{n+1}|$$

such that $|\perp_i|$ is obtained from $|\perp_{i-1}|$ by substitution of words. So each path proposes a sequence of substitutions also.

Let p be a path in $\mathcal{G}$. An arrow x_m is *directly connected.* with a connector $\perp_k$ in p iff the participation $\perp_m$ of $cod(x_m)$ in $|\perp_m|$ is a conveyed (by substitutions of p) participation of $cod(x_m)$ in $|\perp_k|$; an arrow x_m is directly connected with another arrow x_k in p iff it is directly connected with $\perp_{k+1}$ and the center of $\perp_m$ is a conveyed participation of such a subword of $|\perp_{k+1}|$ that intersects with the center of $\perp_k \bullet dom(x_k)$. If x_1 and x_m are directly connected with $\perp_k$ in p the connection with x_m *is to the left* of the connection with x_m iff the predecessors of $\perp_1$ and $\perp_m$ in $|\perp_k|$ do not intersect and the first is situated to the left of the second.

A path in $\mathcal{G}$ is called canonical iff it satisfies the following conditions:

- each arrow situated between two arrows directly connected with a connector is directly connected with the connector also;

- if two arrows x_1 and x_m are directly connected with a connector and the connection with the first is to the left of the connection with the second then the first is situated in the path to the left of the second;

- there are not two connected forks in the path.

An operation "transposition" can be applied to any two adjacent unconnected arrows x_{m-1} and x_m in a path

$$p = \langle \perp_1, x_1, \perp_2, x_2, \ldots, \perp_{m-1}, x_{m-1}, \perp_m, x_m, \ldots, \perp_n, x_n, \perp_{n+1} \rangle$$

to transform it to another path

$$q = \langle \perp_1, x_1, \perp_2, x_2, \ldots \top_{m-1}, x_m, \top_m, x_{m-1}, \ldots, \perp_n, x_n, \perp_{n+1} \rangle$$

where $\top_{m-1}$ is obtained from $\perp_{m-1}$ removing its center over the predecessor of the center of $\perp_m$ and $\top_m$ is obtained from $\perp_{m-1}$ substituting the predecessor of the center of $\perp_m$ by $dom(x_m)$.

The operation is convertable. It is easy to spread it over any two arrows in a path which are not adjacent but fulfil the condition that there are no arrows between them, the second one is connected with.

Similarly, an operation "coalescence of forks" can be introduced. It includes applying of transpositions so to put all connected forks in neighbourhood, replacement of each group of k forks by ∇^{k+1} and the corresponding modifications of connectors.

PROPOSITION 1. Each path in a polygraph $\mathcal{G}$ with a free monoid of objects can be transformed to a unique canonical path applying finitely many times the operations transposition and coalescence of forks.

Let $p_1 = \langle \perp_{1,1}, x_{1,1}, \perp_{1,2}, x_{1,2}, \ldots, \perp_{1,k}, x_{1,k}, \perp_{1,k+1} \rangle$ and $p_2 = \langle \perp_{2,1}, x_{2,1}, \perp_{2,2}, x_{2,2}, \ldots, \perp_{2,\ell}, x_{2,\ell}, \perp_{2,\ell+1} \rangle$ be two paths in a polygraph $\mathcal{G}$ and $\top = u \underline{cod^*(p_2)} v$ be a connector such that $|\top| = dom^*(p_1)$. A *composition* of p_1 and p_2 by $\top$ is a path

$$p_1 \overline{\top\top}' p_2 = \langle \top_1, x_{1,1}, \top_2, x_{1,2}, \ldots, \top_k, x_{1,k}, \top_{k+1}, x_{2,1}, \top_{k+2}, x_{2,2},$$

$$\ldots, \top_{k+1}, x_{2,\ell}, \top_{k+\ell+1} \rangle,$$

where $\top_i = \bot_{1,i}$ for $i=1,2,\ldots,k$, $\top_{k+j} = \top \bullet \bot_{2j}$ for $j=1,2,\ldots,l$ and $\top_{k+\ell+1} = \top \bullet \bot_{2,\ell+1}$. A *canonical composition* of p_1 and p_2 by $\top$ is a path $p_1 \overline{\top}\, p_2$ obtained from $p_1 \overline{\top\top}' p_2$ after transforming it into canonical form.

Each path in a polygraph $\mathcal{G}$ with a free monoid of objects can be represented as a composition of elementary paths.

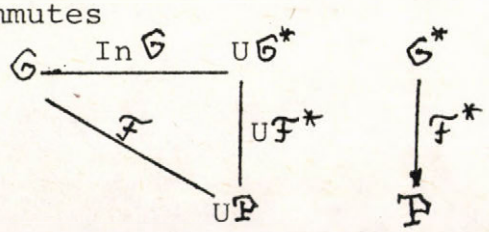
The λ*-polycategory of paths* over a polygraph $\mathcal{G}$ with a free monoid of objects is represented by the string

$$\mathcal{G}^* = \langle \mathbb{V}^*, \mathbb{P}, +, \bullet, \mathrm{dom}^*, \mathrm{cod}^*, \gamma, I \rangle$$

where $\mathbb{P}$ is the set of all canonical paths in $\mathcal{G}$, dom* and cod* put in correspondence the beginning and the end to each path respectively, γ is the canonical composition of paths and $I: \mathbb{V}^* \to \mathbb{P}$, $I: u \mapsto p_u$. One can easy see that it is a λ-polycategory with forks.

Let us denote by $U\mathcal{P}$ the polygraph obtained from a polycategory $\mathcal{P}$ after forgetting the composition law for the arrows and by $\mathrm{In}_{\mathcal{G}} \doteq \langle \mathrm{In}_{\mathcal{G}_0}, \mathrm{In}_{\mathcal{G}_1} \rangle$ the inclusion morphism $\mathcal{G} \to U\mathcal{G}^*$ determined by the correspondences $\mathrm{In}_{\mathcal{G}_0}: v \mapsto v$, $\mathrm{In}_{\mathcal{G}_1}: x \mapsto p_x$.

PROPOSITION 2: $\mathcal{G}^*$ is a free λ-polycategory with forks over the polygraph $\mathcal{G}$ with free monoid of objects in the sense that for each λ-polycategory $\mathcal{P}$ and each polygraph morphism $\mathcal{F} = \langle \mathcal{F}_0, \mathcal{F}_1 \rangle$ $\mathcal{F}: \mathcal{G} \to U\mathcal{P}$ there exists a unique functor (λ-polycategory morphism) $\mathcal{F}^*: \mathcal{G}^* \to \mathcal{P}$ such that the following diagram commutes



2. PARALLEL PROGRAMS

2.1. Let $G = \langle \mathbf{V}^*, \mathbb{A}, +, \bullet, dom, cod \rangle$ be a polygraph with a free monoid of objects $\langle \mathbf{V}^*, +, \bullet \rangle$ over a set of nodes $V = \{v_1, v_2, \ldots\}$

A sequence $s \doteq x_1, x_2, \ldots, x_m$ of arrows such that $cod(x_i)$ and $dom(x_{i+1})$, $(i=1,2,\ldots,m-1)$ contain common nodes will be called a *string* of arrows.

An object $u \in \mathbf{V}^*$ is in *conformity* with another object $v \in \mathbf{V}^*$ iff for any two strings of arrows s_1, s_2 which lead from v to u the stituation of the codomains of the last arrows of s_1 and s_2 in u (to the left or to the right with possible overlapping) is the same as the situation of the domains of the first arrows of s_1 and s_2.

Two arrows x_1 and x_2 such that $dom(x_1)$ and $dom(x_2)$ contain common nodes will be called alternative arrows.

A polygraph G with two selected objects - a *start object*, u_s and an *end object*, u_e will be called a *program scheme* iff it satisfies the following conditions:

(i) - $\mathbb{A}$ and $\mathbf{V}$ are finite sets;

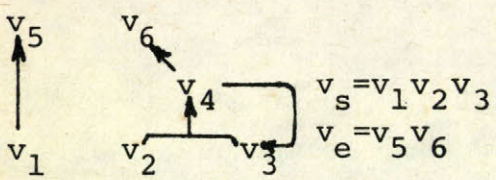
(ii) - if the domains and/or codomains of two arrows contain common nodes their participations form a subword of any of the domains or codomains under consideration;

(iii) - the end object is in conformity with the start object;

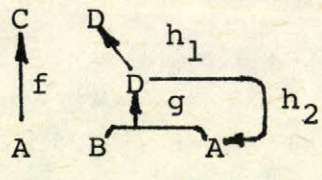
(iv) - any two strings of arrows which begin in the start object and finish with arrows with common nodes in codomains branch out through alternative arrows or one is contained in the other;

(v) - all the nodes which take part in the domain of some arrow but do not take part in the codomain of any arrow take part in the start object; there is no arrow such that a node which takes part in the end object takes part in its domain also.

A simple example of a (parallel) program scheme, represented in diagram form is shown on the next figure:



(a) a program scheme $\mathcal{G}$



(b) a program in the shape $\mathcal{G}$

Let $\mathcal{C}$ be any subcategory with forks of the polycategory Pfn such that all the functions in it are computable. By $U\mathcal{C}$ as usual we denote the corresponding polygraph obtained after forgetting the composition law for the arrows in $\mathcal{C}$.

A (deterministic) *program* in the shape of the program scheme $\mathcal{G}$ (or an interpretation of $\mathcal{G}$) is a functor $P: \mathcal{G} \to U\mathcal{C}$ such that the P-images of any two alternative arrows are functions whose ranges of definition (in the corresponding to the common nodes particular domains) are disjoint.

2.2. Following the ideas similar to that used in the Theory of Petry nets a concept of data flow through a program scheme $\mathcal{G}$ can be introduced as a token game:

- at the beginning (step 0) all the nodes in the start object are marked with active tokens;
- at the step t each arrow, which is not alternative with any other, with marked domain such that at least one of its nodes is marked with an active token can be activated to convert the tokens in its domain to passive and to give raise to active tokens in all nodes in its codomain; in alternative situations just one arrow can be selected to act as described;
- the data flow can be stopped at some step or it can stop naturally if there is no possibility for it to be continued.

Given an object u which is in conformity with the start object a data flow is a *data flow to u* if when it stops all the nodes in u are marked with active tokens and there are no other active tokens.

PROPOSITION 3: There is no data flow in a program scheme such that:

(i) - a node is supplied with an active token by two different arrows simultaneously;

(ii) - a node marked with an active token is supplied with an active token again before its dead.

Proof: Suppose the opposite for a node. In both cases following back the movement of the active tokens two strings s_1 and s_2 can be formed such that: their arrows have been activated; they begin from the start object; their last arrows have the node in common. They cannot branch out through alternative arrows. So one of them contains the other. In the case (i) inclusion is not possible (we want the node to be marked with an active tokens simultaneously). So s_1 and s_2 coincide. In the case (ii) because of the inclusion the token should be made passive before being forced to be active again.

An element of a set $A_1xA_2x...xA_m$ will be called a *string of data* also. Each string of data $<a_1,a_2,...,a_m>$ has $a_1,a_2,...,a_m$ as components.

Given a string of data from the start set a *calculation according to a data flow* can be defined as follows:

- at the step 0 a string of data from the start object is available;

- at the step t a function is calculated for the available data iff it corresponds to an arrow which is activated in the data flow and the available data are in the domain of the function; in this case new avaliable data appear in its codomain according to (ii) the available data for the step t can be arranged in appropriate string of data.

A data flow (to u) is *adequate* to a string of data from the start set iff all the functions of the corresponding calculation can be calculated (in particular from two alternative arrows the arrow which corresponds to a function defined for the available data should be activated). A data flow (to u) is *punctual* for a string of data from the start set iff it is adequate to the string of data and the calculation can not be continued according to another adequite data flow (to u) containing the data flow under consideration.

Given a program P and an object u which is in conformity with the start object a *main calculation to u* over some string of data from the start set according P is a calculation according to a punctual (to the string of data) data flow to u in P.

For each string of data from the start set there is just one punctual data flow in P, at most one punctual data flow to u in P and so at most one main calculation to u.

A *result* of a calculation (to u) according to a given program P for the given string of data from the start set is a string of data received in the end set (the set corresponding to u) according to the main calculation to the end object (to u) if there is any and otherwise there is no result.

Propositin 3 allows us to state:

PROPOSITION 4: There is just one result of a calculation, if any, for a given string of data from the start set according to a program P.

The *function defined by calculation* according to a program is a function from the start set to the end set of the program which puts in correspondence to each string of data the result of the calculation over it. The *operational semantics* for programs supplies a program with the function defined by calculation as meaning.

2.3. The considerations implemented above intuitively suggest paths in a program scheme as formal descriptions of data flows.

Let $P: \mathcal{G} \to U\,\mathcal{C}$ be a program. According to Proposition 2 P has a unique extension to a functor $P^*: \mathcal{G}^* \to \mathcal{C}$ from the λ-polycategory with forks of paths over $\mathcal{G}$. The set of the arrows in $\mathcal{G}$ is partially ordered by a relation "=":

$p \subseteq q$ iff p is an initial part of q.

Given a program $P: \mathcal{G} \to U\mathcal{C}$ the relation $\langle P^*(v_s), P^*(v_e), \delta \rangle$ where

$$\delta = \left\{ \langle a, [P^*(q)](a) \rangle \;\middle|\; \begin{array}{l} q \text{ is a maximal (related to } \subseteq) \text{ path in } \mathcal{G} \\ \text{such that } P^*(q) \text{ is defined for a, } dom^*(q) = \\ = u_s, cod^*(q) = u_e \end{array} \right\}$$

will be called the *conceptual meaning* of P. The semantics which supplies each program with its conceptual meaning will be called *conceptual semantic*. (We use the term "conceptual" to distinguish between conceptual semantic, operational semantic and denotational semantic, the latter usually connected with fix-point constructions.)

PROPOSITION 5: Conceptual and operational semantics coincide.

Proof: Each path q from the start object to an object u which is in conformity with it describes a data flow. Looking on it from the right to the left it can be considered as a sequence of objects and arrows activated in consequent steps. According to the superposition rules in $\mathbb{P}$fn if $P^*(q)$ is defined for a $P^*(u_s)$ then the calculation according these data flow gives as a result $[P^*(q)](a)$ in $P^*(u_e)$. (These statements can be proved more strictly by induction.)

Each (finite) data flow to an object u which is in conformity with the start object can be described as a (canonical) path q from the start object to u such that if the data flow is adequate to a string of data a then P*(q) is defined for a and the result of the calculation according to the data flow over it is [P*(q)](a).

To prove this statement we shall use induction on the number of steps in a data flow.

Suppose that all data flows of t steps fulfil the statement. Consider a data flow of t+1 steps to an object u which is in conformity with the start object. Let at the step t+1 the arrows $x_1, x_2, \ldots, x_n$ have been activated. Their codomains take part in u and do not overlap. If the order of situation of codomains coincides with the chosen order of arrows the object u is of the form

$$u = u_1 cod(x_1) u_2 cod(x_2) \ldots u_n cod(x_n) u_{n+1}.$$

All the nodes in it are active and the nodes in codomains have been activated after the step t+1. This leads us to a conclusion that the object

$$w = u_1 dom(x_1) u_2 dom(x_2) \ldots u_n dom(x_n) u_{n+1}$$

is in conformity with the start object (suppose the opposite and u will not be in conformity with the start object) and all the nodes in it have been activated after the step t. Restricting the data flow under consideration a data flow to the object w of t steps can be obtained. By induction hipothesis there is a path p fulfilling the statement for this data flow.

Consider the path

$$q = q' \circ p = \langle (u_1 \underline{cod(x_1)} u_2 cod(x_2) \ \ldots \ u_n cod(x_n) u_{n+1}), x_1,$$
$$(u_1 dom(x_1) u_2 \underline{cod(x_2)} \ \ldots \ u_n cod(x_n) u_{n+1}), x_2,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$
$$(u_1 dom(x_1) u_2 dom(x_2) \ \ldots \ u_n dom(x_n) u_{n+1}) \rangle \circ p.$$

Since $P^*(q) = P^*(q') \circ P^*(p)$, according to the rules of superposition in Pfn the evaluatin of $P^*(q)$ for the string of data a can be done continuing the evaluation of $P^*(p)$ in the manner described by q'. So the result of executing of t+1(th) step of the calculation according to the given data flow on the result of the calculation till the step t will be obtained. That is just the result of the calculation according to the given data flow.

Moreover, if a data flow is punctual for a given string of data then the corresponding path is a maximal path defined for it (suppose the opposite and the data flow could be continued to another data flow to the same object adequate to the given string of data) and vice verse.

These statements related to the end object prove the proposition.

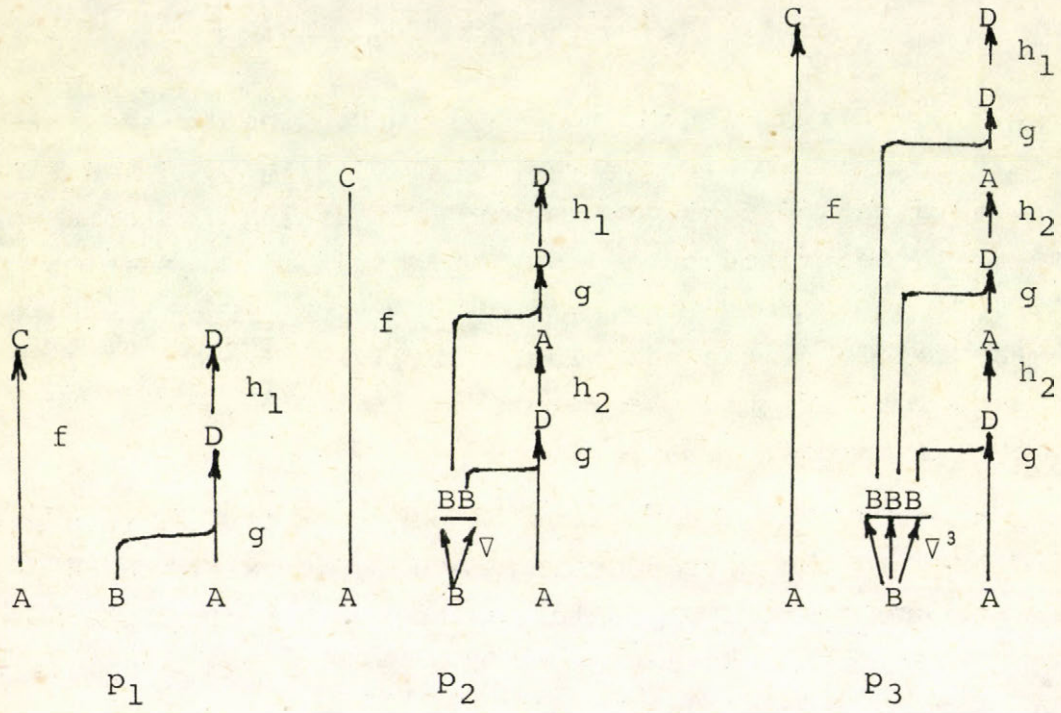
COROLLARY: The conceptual meaning of a program is a function.

The conceptual meaning of a program $P: G \rightarrow U C$ can also be defined as a relation $\langle P^*(u_s), P^*(u_e), \delta \rangle$ where

$$\delta = U\{P^*(q) \mid dom^*(q) = u_s, \ cod^*(q) = u_e\}$$

The limitations of the objects which could be end objects insure the maximality of the paths. So the semantic is the same. But such a definition supplies other objects with relations instead of functions as meaning.

As an example, on the next figure the interpretations of three paths to $u_e=v_1v_2$ for the program scheme and the program presented above are shown in diagram form.



3. CONCLUDING REMARKS

There are many other problems, like the problem of termination or the problem of equivalence which have not been discussed in the present paper. The reader could try to consider them following the papers [HCT, JCS].

The presented constructions are not perfect in several directions. First of all, there are two many restrictions on a polygraph to be a program scheme. Some of them can be avoided if more complicated polycategories (we shall need projections and transpositions [P]) are used. Moreover, it is desirable to find out "external" characteristics of some notions as "domain of definition" or "meaning of a program" instead of the elementwise used here. Such characteristics will give possibility

to involve other polycategories on the place of $\mathbb{P}$fn.

REFERENCES

JCS - J.A.Goguen, J.W.Thatcher, E.G.Vagner, J.B.Wright, "A Junction between Computer Science and Category Theory, I,II" IBM Watson Res. Reports, 1973.

HCT - J.A.Goguen, "On Homomorphisms, Correctness, Termination, Unfoldments, and Equivalence of Flow Diagram Program", JCSS 8, 1974.

P.EP - Y.Velinov, "Polycategories. Elementary Properties" University Annual - Applied Mathematics, tome 17, book 1, Technica, Sofia, 1981.

PSMC I - Y.Velinov, "Polycategories and Strict Monoidal Categories I", University Annual - Applied Mathematics, tome 17, book 4, Technica, Sofia, 1981.

PSMC II - Y.Velinov, "Polycategories and Strict Monoidal Categories II", University Annual - Applied Mathematics, Technica, Sofia, 1984.

P - Y.Velinov, "Polycategories", Proc. "Second Symposium N-ary Structures", Varna, 1983.

CFP - Y.Velinov, "A Construction of Free λ-polycategories", Proc. "Second Symposium N-ary Structures", Varna, 1983.

PR - Y.Velinov, "Polycategories and Recursiveness", Proc. "First Symposium N-ary Structures", Skopje, 1982.

CAP - Y.Velinov, "Combinatorial Arrows in a Polycategory", Proc. "First Symposium N-ary Structures", Skopje, 1982.

Hálók, polikategóriák és a párhuzamos programok szemantikája

Y.P. Velinov

Összefoglaló

A szerző bizonyos összefüggésekre mutat rá, amelyek a Petri hálók és polikategóriák között van. Ezen összefüggések alapján a programok szemantikájának kategória-elméleti tárgyalását kiterjeszti a párhuzamos programokra.

Сети, поликатегории и семантики параллельных программ

И. П. Велинов

Р е з ю м е

В статье устанавливается связь между сетями Петри и поликатегориями. На основе этой связи расширяется категорический подход к семантике программ на более широкий класс параллельных программ.

MTA SZTAKI Közlemények 33/1985 p. 113-143

ALGORITHMS FOR FINDING MINIMAL KEYS AND ANTIKEYS OF RELATIONAL DATA BASES

VU DUC THI

Computer and Automation Institute
Hungarian Academy of Sciences

INTRODUCTION

At present, we often design the data base management systems in many various ways. We have three main datamodel types: hierarchical, network and relational. The relational datamodel, which was defined by E.F. Codd [2], is one of the possible datamodels of a data base. In this datamodel the form of datastorage is a matrix.

The minimal keys and antikeys play important roles for the logic and structural investigation of this model. In particular the role of antikeys for the 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 the minimal keys.

In [3], the equivalence of minimal keys with Sperner--systems was proved. A set of anti keys is also Sperner system.

The main purpose of this paper is to give the representation of minimal key in the set of antikeys and investigate, evaluate the following algorithms:

- Five algorithms, each of which find one minimal key /two of them are linear/.
- Algorithm find the set of all minimal keys.
- Algorithm find the set of all anti keys.

1. DEFINITIONS AND THE REPRESENTATION OF MINIMAL KEY

Let $\Omega = \{1,\ldots,n\}$ and $P(\Omega)$ its power set.
The mapping $F: P(\Omega) \to P(\Omega)$ is called a closure operation over Ω, iff for every $A,B \in P(\Omega)$:

(1.1) $A \subseteq F(A)$ (extensivity),

(1.2) $A \subseteq B \Rightarrow F(A) \subseteq F(B)$ (monotonity),

(1.3) $F(F(A)) = F(A)$ (idempotency).

Let M be an m x n matrix and Ω be the set of its columns. Let $F_M(A), (A) \subseteq \Omega$, be a function such that $F_M(A)$ contains the i-th column of M iff any two rows identical in columns be longing to A are also equal in the i-th column. It is easy to see that $F_M(A)$ is a closure operation. We say that M represents the closure operation F if $F = F_M$. It is known ([1]) that any closure operation is representable by an appropriate matrix M.

Let F be a closure operation over Ω and $A \subseteq \Omega$. A is a key of F if $F(A) = \Omega$. A is a minimal key of F if A is a key, but $F(B) \neq \Omega$ for any profer subset B of A. Denote K the set of the minimal keys. It is known ([3]) that K is the set of minimal keys of any closure operation f and only if K is a non-empty Sperner-system. That is: $A,B, \in K: A \not\subseteq B$.

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

$$K^{-1} = \{A \subset \Omega : (B \in K) \Rightarrow (B \not\subseteq A) \text{ and } (A \subset C) \Rightarrow (\exists B \in K)(B \subseteq C)\}.$$

It is easy to see that the antikeys of K are the subsets of Ω not containing the elements of K and which are maximal for this property and K^{-1} is also a Sperner-system.

In this paper if one Sperner-system play the role of the set of minimal keys (antikeys), then always we assume that this Sperner-system is not empty (not contains Ω).

Theorem 1.1 ([1,4]). If K is an arbitrary Sperner-system, then there is a closure operation $F(F')$ for which $K = K_F$ $(K = K_{F'}^{-1})$.

Let F be an arbitrary Sperner-system over Ω. Denote $Z(F) = \{A:F(A) = A\}$ and $T(F) = \{A\subset\Omega:F(A) = A$ and $\bar{\exists}B \in Z(F)\setminus\{\Omega\}: A \subset B\}$. The elements of $Z(F)$ are called closed sets. It is clear that $T(F)$ is the family of maximal closed sets. We have a following lemma:

Lemma 1.2.([5]). $K_F^{-1} = T(F)$.

Lemma 1.3. Let K be a Sperner-system over Ω and K^{-1} is the set of antikeys of K. Then if $A \in K$ (that is: A is a minimal key), then there is a $B \in K^{-1}$, and an $a \notin B$ $(a \in \Omega)$ such that $A \subseteq B \cup \{a\}$.

Proof: If $|A| = 1$ then it is clear that $\forall B \in K : A \cap B = \emptyset$. Hence $A \subseteq B \cap A$ (B is an arbitrary element of K^{-1}).

If $|A| \geq 2$ then, by Theorem 1.1 there exists a closure operation F such that $K = K_F$. Let D be the set for which $D \subset A$ and $A \setminus D = a (a \in \Omega)$. By Lemma 1.2 and from $A \in K$, we have $F(D) \neq \Omega$, and there exists a $B \in K^{-1}$ such that $F(D) \subseteq B$. It is clear that $A \subseteq B \cup \{a\}$. A lemma is proved.

It is obvious that for all B $(B \in K^{-1})$ there is an A $(A \in K)$ such that $A \subseteq B \cup \{a\}$, where $a \notin B$.

Lemma 1.4. Let K be a Sperner-system over Ω and $K^{-1} = \{B_1,\ldots,B_m\}$ is the set of antikeys of K, $A \subseteq \Omega$. Then $A \in K$ (that is: A is a minimal key) if and only if $A \not\subset B_i$ $(\forall_i :1 \leq i \leq m)$ and for every D $(D \subset A)$, there is a $B_i (B_i \in K^{-1})$ such that $D \subseteq B_i$.

Proof. If there exists a $B_i \in K^{-1}$ such that $B_i \subset A$, then A is a key. If $K^{-1} \cup A$ is a Sperner-system, then from Theorem 1.1 there exists a closure operation F such that $K = K_F$. Consequently, if $F(A) \neq \Omega$, then by lemma 1.2 there is a B_i $(B_i \in K^{-1})$ such that $F(A) \subseteq B_i$, i.e. $A \subseteq B_i$. This contradicts with the fact that $A \not\subseteq B_i$ $(\forall i : 1 \leq i \leq m)$.

Hence A is a key. It is clear that by the definition of minimal key, A is a minimal key.
On the other hand, we suppose that A is a minimal key. It is clear that $A \not\subseteq B_i$ $(\forall_i : 1 \leq i \leq m)$, and for every D $(D \subset A)$, if $D \not\subseteq B_i (\forall_i : 1 \leq i \leq m)$ then from the above proof D is a key, which conflicts with the fact that A is a minimal key. Consequently, there is a B_i $(B_i \in K^{-1})$ such that $D \subseteq B_i$.
A lemma is proved.

Let $Q = \{A_1,...,A_m\}$ be a family of non-empty sets over Ω and $|A_i| = t_i$ $(\forall i : 1 \leq i \leq m)$. The following numeration of Q was called primary one:

$$A_1 = \{a_1^1, a_2^1,...,a_{t_1}^1\},..., A_i = \{a_1^i,...,a_{t_i}^i\},...,A_m=\{a_1^m,...,a_{t_m}^m\}.$$

Where we choose a_1^1 as follows.

1) Choose a first element a of A_i. If there is an $A_i (1 \leq i \leq m)$ much that $A_i = \{a\}$, then $a_1^1 = a$. Conversely, we mark all elements a, which occur in A_i-s $(1 \leq i \leq m)$ by the star and

2) we choose a following element a of A_1. If there is an $A_i (1 \leq i \leq m)$ such that a is a unique element of A_i which was not marked by the star, then $a_1^1 = a$. Conversely, we mark all elements a, which occur in A_i-s $(1 \leq i \leq m)$ by the star and

3) the 2-th step is repeated until a a_1^1 was chosen.

   - In A_p, where p run from 2 to m, we perform as follows:

     for every p, if there is an a_1^q such that $q < p$ and $a_1^q \in A_p$, then $a_1^p = a_1^q$.

     Conversely,

4) we choose a first element a of A_p, which was not marked by the star. If there exists an A_i $(p \leq i \leq m)$ such that a is a unique element of A_i, which was not marked by

the star, then $a_1^p = a$. Conversely, we mark all elements a, which occur in A_i-s $(p \leq i \leq m)$ by the star and.

5) the 4-th Step is repeated until a_1^p was chosen.

Example 1.5: Let $\Omega = \{1,2,3,4,5,6\}$

$A_1 = \{1,3,4,5\}$, $A_2 = \{1,2,3,4,6\}$ and $A_3 = \{3,4,5\}$ we have

$\{1^*,3,5,6\}$, $\{1^*,2,3,4,6\}$, $\{3,4,5\}$

$\{1^*,3^*,5,6\}$, $\{1^*,2,3^*,4,6\}$ $\{3^*,4,5\}$

$\{1^*,3^*,5^*,6\}$, $\{1^*,2,3^*,4,6\}$, $\{3^*,4,5^*\}$. Consequently,

$A_1 = \{6,1,3,5\}$, by $6 \in A_2$, we have $A_2 = \{6,1,2,3,4\}$, and it is clear that $A_3 = \{4,3,5\}$.

Theorem 1.6: Let $K = \{B_1,\ldots,B_m\}$ be a Sperner-system over Ω, and denote H the set, which $H^{-1} = K$. Then $A \in H$ (i.e. A is a minimal key) if and only if $A = \{a\}$, where $a \notin \bigcup_1^m B_i$ or $A = \bigcup_{i=1}^{\ell} \{a_1^{\Pi_i}\} \cup \{a\}$, where $(\Pi_o,\Pi_1,\ldots,\Pi_\ell)$ is an ordered subset of the indices $1,2,\ldots,m$.

$\{B_{\Pi_o} \cap \bar{B}_{\Pi_i}\} = \{a_1^{\Pi_i},\ldots,a_{\Pi_i}^{\Pi_i}\}$, $1 \leq i \leq \ell\}$ is a primary numeration and $a \notin B_J (\forall J : J \neq \{\Pi_1,\ldots,\Pi_\ell\})$, $a \in B_{\Pi_i}$ $(\forall i : 1 \leq i \leq \ell)$.

Proof. It is clear that $\{a\} \in H$ if and only if $a \notin \bigcup_1^m B_i (B_i \in K))$. We assume that $(\Pi_o,\Pi_1,\ldots,\Pi_\ell)$ is an ordered subset of the indices $1,\ldots,m$ such that $\{B_{\Pi_o} \cap \bar{B}_{\Pi_i} = \{a_1^{\Pi_i},\ldots,a_{\tau_{\pi_i}}^{\Pi_i}\}\}$ $(1 \leq i \leq \ell)$ is a primary numeration, and $a \in B_{\Pi_i}$ $(\forall i : 1 \leq i \leq \ell)$, $a \notin B_J (\forall J : J \neq \{\Pi_1,\ldots,\Pi_\ell\})$ and $A = \bigcup_1^{\ell} \{a_1^{\Pi_i}\} \cup \{a\}$. It is easy to see that $A \not\subseteq B_{\Pi_i}$ by $A \cap (B_{\Pi_o} \cap \bar{B}_{\Pi_i}) \neq \emptyset$ $(\forall i : 1 \leq i \leq \ell)$ and from $a \notin B_J$ we

have $A \not\subseteq B_\tau (\forall B \in K \setminus \{B_{\Pi_1},\dots,B_{\Pi_\ell}\})$. Consequently, $A \not\subseteq B_i$ $(\forall i : 1 \le i \le m)$. It is cleat that $\bigcup_{i=1}^{\ell} \{a_1^{\Pi_i}\} \quad B_{\Pi_0}$.

By the definition of the primary numeration, for every Π_i $(1 \le i \le \ell)$ there is a Π_τ such that $A \cap (B_{\Pi_0} \cap \bar{B}_{\Pi_\tau}) = \{a_1^{\Pi_i}\}$. Hence if D is a profer subset of A, and we suppose that $a \in A \setminus D$, then $D \subseteq B_{\Pi_0}$. If we suppose that $a_1^{\Pi_i} \in A \setminus D$ and $a \in D$ $(1 \le i \le \ell)$, then there exists a Π_τ such that $A \cap (B_{\Pi_0} \cap \bar{B}_{\Pi_\tau}) = \{a_1^{\Pi_i}\}$. Consequently from $a \in D$ and by $\bigcup_{p=1}^{\ell} \{a_1^{\Pi_p}\} \subseteq B_{\Pi_0}$ we have $D \subseteq B_{\Pi_\tau}$. Hence from Lemma 1.4.

A is a minimal key. Conversely, we suppose that A is a minimal key. By Lemma 1.3 there exists a B_i $(B_i \in K)$ such that $A \subseteq B_i \cup \{a\}$, where $a \notin B_i$. Denote $\{B_{\Pi_1},\dots,B_{\Pi_\ell}\} = \{B_\tau \in K : a \in B_\tau\}$ and $\Pi_0 = i$. It is clear that $(\Pi_0,\dots,\Pi_\ell)$ is an ordered subset of the indices $1,\dots,m$. By K is a Sperner-system, we have $B_{\Pi_0} \cap \bar{B}_{\Pi_i} \neq \emptyset$, and it is obvious that $a \notin B_\tau$ $(\forall \tau : \tau \notin \{\Pi_1,\dots,\Pi_\ell\})$. It is clear that from A is a minimal key and by Lemma 1.4, for every b $(b \in A \setminus a)$ there is a Π_i such that $A \cap B_{\Pi_0} \cap \bar{B}_{\Pi_i} = \{b\}$. It is obvious that $A \cap B_{\Pi_0} \cap \bar{B}_{\Pi_i} \neq \emptyset$ $(\forall i : 1 \le i \le \ell)$. Consequently, we can suppose that for every i such that $((B_{\Pi_0} \cap \bar{B}_{\Pi_i}) \setminus A) \neq \emptyset$, and the first time, $B_{\Pi_0} \cap \bar{B}_{\Pi_i}$ has a following form:

$B_{\Pi_0} \cap \bar{B}_{\Pi_i} = \{a_1^i,\dots,a_{\tau_i}^i\}$ and $A \cap B_{\Pi_0} \cap \bar{B}_{\Pi_i} =$

$= \{a_\tau^i, a_{\tau+1}^i,\dots,a_{\tau_i}^i\}$ where $1 < \tau$. Thus, the elements of A stand in the last places of $B_{\Pi_0} \cap \bar{B}_{\Pi_i}$. After we construct a primary numeration of $\{B_{\Pi_0} \cap \bar{B}_{\Pi_i}, \quad 1 \le i \le \ell\}$.

Denote $C = \bigcup_{i=1}^{\ell} \{a_1^{\Pi_i}\} \cup \{a\}$. It is obvious that $a \in A$. It is clear that from the construction of the primary numeration, we have $a_1^{\Pi_i} \in A$, hence $C \subseteq A$. On the other hand from the proof the necessary condition C is a minimal key. Consequently, $C = A$. The theorem is proved.

Denote $a_1^{\Pi_o}$ the element a. We have a following representation of a minimal key: $A = \bigcup_{i=o}^{\ell} \{a_1^{\Pi_i}\}$, where $\ell \geq 0$.

2. ALGORITHMS FIND ONE MINIMAL KEY

In this section, we investigate, evaluate five algorithms which find one minimal key.

For the evaluating of the time complexity of algorithms, we always suppose that the elementary step being counted is the comparison of two attribute. Thus, we also assume that $\forall A\ (A \subseteq \Omega)$: A is a sorted list of attributes. Consequently, a Boolean operation on two subsets of Ω requires at most $O(n)$ elementary steps, where $|\Omega| = n$.

Let K be a Sperner-system over Ω ($|\Omega| = n$), and $K = \{B_1, \ldots, B_m\}$. Denote $|K| = n \times m$ the length of K.

Now we investigate a first algorithm:

Let K be a Sperner-system, K^{-1} is a family of antikeys of K. $B \in K^{-1}$, $a \in \Omega \setminus B$, $B = \{b_1, \ldots, b_m\}$ and $G = \{B_\tau \in K^{-1} \quad a \notin B_\tau\}$.

Let $T_o = B \cup \{a\}$.

$$T_{q+1} = \begin{cases} T_q \setminus \{b_{a+1}\} \text{ if } \forall B_i \in K^{-1} \setminus G : T_q \setminus \{b_{a+1}\} \not\subseteq B_i, \\ T_q \quad \text{othwewise.} \end{cases}$$

Theorem 2.1. ([5]). $\{T_o, T_1, \ldots, T_m\}$ are the key and T_m is a minimal key, i.e. $T_m \in K$.

In a similar manner we have:

Let $K = \{B_1, \ldots, B_m\}$ be a Sperner-system, $B = \{b_1, \ldots, b_\ell\}$ is a key.

Denote H the set such that $H^{-1} = K$, then there is an $A \in H$ such that A B.

$$T_{q+1} = \begin{cases} T \setminus \{b_{q+1}\} & \text{if } \forall B_i \in K: T_a \setminus \{b_{a+1}\} \not\subseteq B_i, \\ T_q & \text{otherwise.} \end{cases}$$

By the proof is analogous to the theorem 2.1. we have

Proposition 2.2: If K is a family of anti keys, then $\{T_o, \ldots, T_\ell\}$ are the keys and T_ℓ is a minimal key.
From Proposition 2.2 we construct a following algorithm.

Algorithm 2.3: with a quadratic-time.

INPUT: $K = \{B_1, \ldots, B_m\}$ is a Sperner-system over $\Omega = \{1, \ldots, n\}$ and B is a key (i.e. if we denote H the set such that $H^{-1} = K$, then there is an $A \in H$ such that $A \subseteq B$).

OUTPUT: T is a minimal key (i.e. $T \in H$).

DATA STRUCTURES:

1) $K[1:m]$ is an array, which contains the elements of K.
2) B is a set, which is a key.
3) T_1 is a Boolean variable.
4) T is a workvariable.

```
Procedure  MINIKEY1(K,B):
    T ← B;
    for each  b ∈ B  do
        begin  T1 ← true;
               for  i ← 1  until  m  do
                 if  T\{b} ⊆ K[i]  then  T1 ← false;
                 if  T1  then  T ← T\{b}
        end;
    return T.
```

It is clear that Algorithm 2.3 requires $O(n^2 \cdot m)$ (i.e. $O(n|K|)$ elementary operations, and by Proposition 2.2 we have:

Corollary 2.4: Algorithm 2.3 determines one minimal key, and its the time complexity is $O(n|K|)$.

Let $K = \{B_1,\ldots,B_m\}$ be a Sperner-system and B is a key of H, where $H^{-1} = K$ Let $A_i = B \cap \bar{B}_i$ $(\forall i : 1 \leq i \leq m)$. In A_i, where i run from 1 to m, we perform as follows:

let $C_o = \emptyset$, for each i, if $C_{i-1} \cap A_i \neq \emptyset$, then $C_i = C_{i-1}$ conversely,

1) we choose a first element a of A_i. If there is an A_τ $(i \leq \tau \leq m)$ such that $A_\tau = \{a\}$, then $C_i = C_{i-1} \cup \{a\}$. Conversely we delete all elements a, which occur in A_τ-s $(i \leq \tau \leq m)$ and

2) the first step is repeated until we can not continue the choice.

By the proof is analogous to Theorem 1.6 we have:

Proposition 2.5: C_m is a minimal key (i.e. $C_m \in H$).

By Proposition 2.5 we construct a following algorithm.

Algorithm 2.6: with a quadratic time.

INPUT: $K = \{B_1,\ldots,B_m\}$ is a Sperner-system over $\Omega = \{1,\ldots,m\}$ and B is a key.

OUTPUT: $T \in H$, where $H^{-1} = K$.

DATA STRUCTURES:

1) K[1:m] is an array, which contains the elements of K.
2) M[1:m] is a work array.
3) B is set of attributes, which is a key.
4) C is a workvariable.

```
Procedure MINIKEY2 (K,B):
KEZDET: for  i ← 1  until  m  do
                M[i] ← B∩ (Ω \ K[i]); C ← Ø;
CIKLUS1:    for  i ← 1  until  m  do
               if  C ∩ M[i] = Ø  then
            begin

CIKLUS2:    for each  b ∈ M[i]  do
            begin for  τ ← i  until  m  do
                  if  b = M[τ] then begin C ← C∪{b};
                                    go to UTOLSO end;
                  for  τ ← i  until  m  do
                       M[J] ← M[τ] - {b}
            end;

UTOLSO:  empty
         end;
         return C.
```

It can be seen that in the cases for which there are many $B \cap \bar{B}_i$ such that $|B \cap \bar{B}_i|$ is few, in particular $|B \cap \bar{B}_i|=1$, or $B \cap \bar{B}_i$-s have a common element, the algorithm 2.6 better than the algorithm 2.3.

In KEZDET-section an algorithm requires $O(n \times m)$ elementary operations. It is easy to see that in CIKLUS1-section the algorithm requires $O(n \sum_{i=1}^{\ell} \ell_i(m-\tau_i) + \sum_{i=1}^{\ell+1} \tau_i)$ elementary operations, where ℓ is the number of iterations of CIKLUS2, ℓ_i is the number of attributes, which the algorithm process in the i-th iteration of CIKLUS2 τ_i is the number of $M[q]$-s such that $C \cap [M]q \neq \emptyset$ from the i-1 -th iteration to the i-th iteration of CIKLUS2. $\tau_{\ell+1}$ is the number of $M[q]$-s such that $C \cap M[q] \neq \emptyset$ after the ℓ-th iteration of CIKLUS2. It is obvious that $\tau_1 = 0$.

It can be seen that $\sum_{i=1}^{\ell+1} \tau_i + \ell = m$ and $\sum_{i=1}^{\ell} \ell_i \leq n$.

In the worst-case, it is easy to see that the algorithm requires $O(n^2 \cdot m)$ elementary operations and by the Proposition 2.5 we have:

Corollary 2.7: The algorithm 2.6 determines one minimal key, and its the time complexity is $O(n|K|)$.

Now from the algorithm 2.6 and with the aid of data-structures we construct an algorithm, which find one minimal key and his time complexity is linear (i.e. proportional to the length of K). For each attribute appearing in $B \cap \bar{B}_i$-s, we construct a linked-list for each $B \cap \bar{B}_i$ also we construct a counter.

Algorithm 2.8: With the linear time.

INPUT: $K = \{B_1,\ldots,B_m\}$ is a Sperner system over $\Omega = \{1,\ldots,n\}$ and B is a key.

OUTPUT: $A \in H$, where $H^{-1} = K$.

DATA STRUCTURES:

1) K[1:m] is an array which contains the elements of K.
2) D[1:m,1:n] is a work array.
3) M[1:m] is a work array, which contains $B \cap \bar{B}_i$-s.
4) C[1:m] is an array of counters.
5) L[1:n] is an array of lists.
6) AT1[1:n] is a work array.
7) AT2[1:n] is a work array.
8) A is a work variable.
9) B is a key the set of attributes.
10) D1 is a work variable.

Procedure MINIKEY3(K,B):

```
KEZDET:   A ← O;
          for  i ← 1  until  n  do
               begin  AT1[i]←1; AT2[1]←1; L[i] ← O  end;
          for  i ← 1  until  m  do begin (C[i] ← O;
                                 M[i]←B∩(Ω\K[i]) end;
               for  i ← 1  until  m  do
                    for each  τ ∈ M[i]  do
                     begin C[i]←C[i]+1; D(i,C[i])←τ;←L[τ]∪{i} end+

CIKLUS1:  for  i ← 1  until  m  do
            begin
             for  τ ← 1  until  C[i]  do
             begin if  AT2(D[i,τ]) = O  then go to UTOLSO;
                   if  AT1(D[i,τ]) = 1  then
               begin
```

```
CIKLUS2:  for each  q ∈ L(D[i,τ]) do
           if  C[q]=1  then begin  A←A ∪ {D[i,τ]}; AT2(D[i,τ])←0;
                              go to UTOLSO end;
          for each  q ∈ L(D[i,τ])  do
            begin  C[q]←C[q]-1; AT1(D[i,τ]←0  end end
          end;

UTOLSO:   empty
           end;
         return A.
```

In KEZDET-section we construct C[i]-counters ($C[i]=|B \cap \bar{B}_i|$), L[i]-linked lists and D[i,τ]-s, which contain the $B \cap \bar{B}_i$-s. By $|B \cap \bar{B}_i| \leq n$ the algorithm requires $O(|K|)$ elementary operations in the KEZDET-section.
In the CIKLUS2-section if $a \in A$, then AT2[a] = 0, and if $a \notin A$, then AT1[a] = 0. Consequently, in the worst-case the number of iteration of CIKLUS2 is not greater than n. It is obvious that $|L[i]| \leq m$ $(\forall i : 1 \leq i \leq n)$. Hence in the CIKLUS2-section the algorithm requires $O(n \cdot m)$ elementary operations. If AT1[a] = 1 ∧ AT2[a] = 1 is true, then CIKLUS2 workes, consequently, in the CIKLUS1-section the algorithm also requires $O(|K|)$ elementary operations. That is the algorithm 2.8 has a time complexity, which is proportional to the length of K.

From Proposition 2.6 we have

Corollary 2.9. The algorithm 2.8 determines one minimal key and its time complexity is $O(|K|)$.

Let $\Omega = \{1,\ldots,n\}$ and $P(\Omega)$ its power set, $G \subseteq P(\Omega)$. we define MIN(G) as follows:

MIN(G) = A, where A is an element of G such that $|A| = \min\{|A_i| : A_i \in G\}$.

If $A \subseteq \Omega$ and $A \neq \emptyset$, then $\nu(A) = a$, where a is an arbitrary element of A.

Let K be a Sperner-system over $\Omega = \{1,\ldots,n\}$ and $K^{-1} = \{B_1,\ldots,B_m\}$ is the set of anti keys of K and B is a key. Denote $G_1 = \{B \cap \bar{B}_1,\ldots,B \cap \bar{B}_m\}$. By Lemma 1.1 $B \cap \bar{B}_i \neq \emptyset$ $(\forall i : 1 \leq i \leq m)$. Let $B \cap \bar{B}_i = B_i^1$, that is: $G_1 = \{B_1^1,\ldots,B_m^1\}$. It is easy to see that can occur $i,\tau (i \neq \tau)$: $B_i^1 = B_\tau^1$. Then we consider them as two different elements.

Let $m = \tau_1$.

For every i $(i \geq 1)$ we define as follows:

$$F_{i+1} = \{B_\tau^i \in G_i : \mathfrak{V}(\mathrm{MIN}(G_i)) \notin B_\tau^i\} \quad \text{and}$$

$$G_{i+1} = \{B_\tau^i \setminus \mathrm{MIN}(G_i) : B_\tau^i \in F_{i+1}\} = \{B_1^{i+1},\ldots,B_{\tau+1}^{i+1}\}.$$

From $|\Omega|$ is finite, it is cleat that there is a natural member p such that $G_p \neq \emptyset$ but $F_{p+1} = \emptyset$.

It can be seen that $p \leq \min(n,m)$ and $\mathfrak{V}(\mathrm{MIN}(G_i)) \neq \mathfrak{V}(\mathrm{MIN}(G_\tau))$, where $1 \leq i \leq p$, $1 \leq \tau \leq p$. $i \neq \tau$.

By $|\mathrm{MIN}(G_i)| \leq |B_\tau^i|$ and if $F_{i+1} \neq \emptyset$ then $B_\tau^i \setminus \mathrm{MIN}(G_i) \neq \emptyset$.

Thus, $G_{i+1} \neq \emptyset$.

Proposition 2.10. $A = \bigcup_{i=1}^{p} \{\mathfrak{V}(\mathrm{MIN}(G_i))\}$ is a minimal key.

Proof. From the definition of G_i $(1 \leq i \leq p)$, for every B_τ^1 $(1 \leq \tau \leq m)$ there exists a F_ℓ such that $2 \leq \ell \leq p+1$, and there is a $B_q^{\ell-1}$ such that $B_q^{\ell-1} \in G_{\ell-1} \setminus F_\ell$, $B_q^{\ell-1} \subseteq B_\tau^1$ (where $1 \leq q \leq \tau_{\ell-1}$). Consequently, $\mathfrak{V}(\mathrm{MIN}(G_{\ell-1})) \in B_\tau^1$. That is $A \cap (B \cap \bar{B}_\tau) \neq \emptyset$

$(\forall J : 1 \leq J \leq m)$. Hence $A \not\subseteq B_\tau$ $(\forall \tau : 1 \leq \tau \leq m)$. Let D be a profer subset of A, if $|A| = 1$, then it is obvious that A is a minimal key.

If there is a $\mathcal{V}(MIN(G_i))$ $(1 \leq i \leq p)$ such that $\mathcal{V}(MIN(G_i)) \in A \backslash D$, then it is clear that there exists a $B \cap \bar{B}_\tau (1 \leq \tau \leq m)$ such that $MIN(G_i) \subseteq B \cap \bar{B}_\tau$. From the definition of G_i and by $A \subseteq B$ we have $A \backslash \mathcal{V}(MIN(G_i)) \subseteq B$. It is obvious that $D \subseteq A \backslash \mathcal{V}(MIN(G_i))$, consequently $D \subseteq B$. By Lemma 1.4 A is a minimal key. The theorem is proved.

Example 2.11. Let $\Omega = \{1,2,3,4,5,6\}$ and $K^{-1} = \{B_1, B_2, B_3\}$, $B_1 = \{1,3,5\}$, $B_2 = \{1,3,4,6\}$, $B_3 = \{2,3,4,5,6\}$. Then let $B = B_3 \cap \{1\}$. It is obvious that $\mathcal{V}(MIN(G_1)) = \{1\}$, $F_2 = \{\{2,4,6\},\{2,5\}\}$. $MIN(G_2) = \{2,5\}$. If $\mathcal{V}(MIN(G_2)) = \{2\}$ then $F_3 = \emptyset$. That is $A_1 = \{1,2\}$ is a minimal key. If $\mathcal{V}(MIN(G_2)) = \{5\}$, then $F_3 = \{2,4,6\}$ and $G_3 = \{4,6\}$. It is clear that if $\mathcal{V}(MIN(G_3)) = \{4\}$, then $A_2 = \{1,5,4\}$ is a minimal key if $\mathcal{V}(MIN(G_3)) = \{6\}$, then $A_3 = \{1,5,6\}$ is a minimal key.

Remark 2.12. Let K be a Sperner-system, $K^{-1} = \{B_1,\ldots,B_m\}$ is the set of anti keys. It is obvious that $B_i \cup \{a\}$ $(a \notin B_i)$ $(\forall i : 1 \leq i \leq m)$ is a key. It is best to denote $\mathcal{V}(MIN(G_1)) = a$ and $G_2 = \{B_i \cap \bar{B} : a \in B \text{ and } B \in K^{-1}\}$.

By Proposition 2.10 we construct a algorithm which find a minimal key. In first time we construct a following algorithm.

INPUT: K is a set of non-negative integers and there exists a k $(k \in K)$ such that $k \neq 0$. p is the number of elements of K, denote $K = \{K[1],\ldots,K[p]\}$.

OUTPUT: i, where $1 \leq i \leq p$, $K[i] = \min\{K[J] \in K : K[J] \neq O\}$.

DATA STRUCTURES

1) K[1 : m] is anarray, which contains the elements of K.
2) A,B are workvariables.
3) M contains the number p which is the number of element of K.

It is clear that $p \leq m$.

Procedure MIN1(K,M):

```
A ← K[1]; B←1;;
for i ← 1 until M do
  if K[i] ≠ O then begin A ← K[i], B i goto T1 end;
T1: for i ← 1 until M do if K[i] ≠ O ∧ K[i] < A
    then begin A ← K[i]; B[i]; B ← i end;
```

Algorithm 2.13. With a quadratic time.

INPUT: $K = \{B_1,\ldots,B_m\}$ is a Sperner-system and B is a key.

OUTPUT: $T \in H$, where $H^{-1} = K$.

1) K[1 : m] is an array which congains the elements of K.
2) B is a set of attributes which is a key.
3) M[1 : m], C[1 : m] are work array.
4) D, D_1, D_2, T, N are work variables.
5) N_1, N_2 are work variables.

Procedure MINIKEY4 (K,B):

```
KEZDET: for i ← 1 until m do begin C[i] ← O;
        M[i] ← B ∩ (𝔐\K[i]) end;
        for i ← 1 until m do
            for each J ∈ M[i] do C[i] ← C[i] + 1;
        T ← Ø; D ← m;
```

```
CIKLUS: while D ≠ O do begin D1 ← D; D2  MIN1(C,D1);
        N ← M(D2);
        select ELEM from N; T ← T ∪ {ELEM}; D ← O;
        for i ← 1 until D1 do
           if ELEM ∉ M[i] then begin D ← D+1; M(D)← M[i] \ N;
           N1 ← M[i]  N; N2 ← |N1|; C(D) ← C[i] -N2
                                    end
                              end;
                        return T.
```

It is easy to see that the algorithm 2.13 is effective, if ELEM is a common element of many B_J^i.

In KEZDET-section the algorithm requires $O(n \cdot m)$ elementary operations. It can be seen that MIN1 requires $O(m)$ elementary operations, where m is the number of the elements of K.

It is obvious that in the CIKLUS-section the algorithm $O(n \sum_{i=1}^{\ell} \ell_i)$ elementary operations, where ℓ is the number of iterations of CIKLUS, ℓ_i is the value of D1 in the i-th iteration of CIKLUS. It is cleat that $\ell \leq n$ and $\ell_i \leq m$ $(\forall i)$

From proposition 2.10 we have:

Corollary 2.14: The algorithm 2.13 determines one minimal key and its time complexity is not greater than $O(n|K|)$, where K is a Sperner-system.

By proposition 2.10, with the aid of linked lists and counters we construct a following algorithm.

Algorithm 2.15: With a linear time.

INPUT: $K = \{B_1,\ldots,B_m\}$ is a Sperner-system over $\Omega = \{1,\ldots,n\}$ and B is a key of H, where $H^{-1} = K$.

```
OUTPUT:  T ∈ H

   1)  K[1:m] is an array, which contains the elements of  K.

   2)  M = {1,2,...,n}  is a sorted list.

   3)  B  is a key.

   4)  M1[1:n],   M2[1:m]  are work array.

   5)  A_J[1:n], L[1:n], C[1:m] are work array.

   6)  T,B1,B2,D1,D2,A1,A2,A3,M3  are work variables.

Procedure MINIKEY5 (K,B):

KEZDET:  T ← O;  for  i ← 1  until  m  do  C[i] ← O.
         for  i   1  until  n  do begin  AT[i] ← 1;
            L[i] ← O end;
         for  i ← 1  until  m  do
            begin  B1 ← B∩(Ω\K[i]); M2[i] ← B1;
              for each  J ∈ B1  do
              begin  C[i] ← C[i]+1;  L[J] ← L[J] ∪ {i}  end
            end;
         for  i ← 1   until  n  do  M1[i] ← M\L[i];
         D1 ← m;  A3 ← O;  B2 ← m;

CIKLUS:  While  D1 ≠ O  do begin  D2 ← MIN1(C,B2); A1 ← 1;
                  for each  J ∈ M2(D2)  fo
                     begin if  AT[J] = O  then goto UTOLSO;
                      if  AT[J] = 1 ∧ A1 = 1  then begin
                      T ← T∪{J};  A1 ← O;
                     A2 ← J;  for each  q ∈ L(D2)  do
                              if  C[q] ≠ O  then begin
                              A3 ← A3+1; C[q] = O  end;
                           goto  UTOLSO end;
                     AT[J] ← O;  M3 ← L[J] ∩ M1 (A2);
                  for each  q   M3  do if  C q  ≠ O
```

```
UTOLSO:  empty
         end;
         D1 ← m - A3
         end;
         return T.
```

It is clear that L[i] $(\forall i : 1 \leq i \leq n)$ is a sorted list, consequently M[i] ← M-L[i] requires O(m) elementary operations. Thus, in the KEZDET-section the algorithm required O(n•m) elementary.

It is clear that, by L[J], M1(A2) are sorted lists, L[J] ∩ M1(A2) requires O(m) elementary operations. From the number of iteration of CIKLUS is not greater than min(n,m) in the CIKLUS-section the algorithm requires O(n•m) (i.e. $O(|K|)$) elementary operations.

By Proposition 2.10 we have:

Corollary 2.16: The algorithm 2.15 determines one minimal key and its time complexity is $O(|K|)$.

Remark 2.17: Let R be a relation over Ω, $R = \{h_1,\ldots,h_m\}$, let $E_{i_J} = \{a \in \Omega : h_i(a) = h_J(a)\}$, where $1 \leq i \leq m$, $1 \leq J \leq m$ and $i \neq J$. Denote $M = \{E_{i_J} : \overline{\exists} E_{s_t} : E_{i_J} \subset E_{s_t}\}$. In [4], E_{i_J} is called the equality set of the relation R. Practically, it is possible that there are many E_{i_J} which equal to each other. We choose one E_{i_J} from M. According to Lemma 1.2 it can be seen that M is the set of anti keys. From give algorithm, which were constructed, by Remark 2.17 we find minimal keys.

Example 2.18: Let $\Omega = \{1,2,3,4,5,6\}$

Relation R:

0	1	0	0	1	0
1	0	1	0	0	1
2	0	0	1	2	2
0	1	2	2	0	3
3	2	1	0	0	0

It can be seen that $M = \{(1,2),(3,4,5),(4,6)\}$, where $E_{14} = \{1,2\}$, $E_{15} = \{4,6\}$ and $E_{25} = \{3,4,5\}$. It is obvious $E_{15} \cup \{1\} = \{1,4,6\}$ is a key. Suppose that we use Algorithm 2.3, then we have:

If $T_o = \{1,4,6\}$, then $\{1,4\},\{1,6\}$ are minimal keys.

If $T_o = \{3\} \cup E_{14}$, then $\{1,3\},\{2,3\}$ are minimal keys.

If $T_o = E_{25} \cup \{2\}$ then $\{2,4\},\{2,5\}$ are minimal keys.

3. ALGORITHMS CONNECTED WITH ALL THE MINIMAL KEYS AND ANTI KEYS

In this section we investigate two algorithms: a first algorithm find all the anti keys and a second algorithm find all the minimal keys.

Now we construct an algorithm, which find the set of anti keys.

Let $K = \{B_1,\ldots,B_m\}$ be a Sperner-system over Ω. We have to construct K^{-1}. For every $t = 1,\ldots,m$ we construct $K_t = \{B_1,\ldots,B_t\}^{-1}$ by induction.

Step t+1: By the inductive hypothesis we have constructed $K_t = \{B_1,\ldots,B_t\}^{-1}$. Construct K_{t+1} as follows:

We suppose that $X_1,\ldots,X_\ell$ are the elements containing B_{t+1} of K_t

That is:

$$K_t = \{X_1,\ldots,X_\ell\} \cup \{A \in K_t : B_{t+1} / A\}.$$

Denote $\{A \in K_t : B_{t+1} / A\}$ by F_t. It can be seen that in sone cases it is possible that $F_t = \emptyset$ and for every t $(1 \leq t \leq m-1)$ we have $\{X_1,\ldots,X_\ell\} \neq \emptyset$.

For every i $(i = 1,\ldots,\ell)$ we construct the anti keys of $\{B_{t+1}\}$ on X_i in the analogous way of as in step 1, which are the maximal subsets of X_i not containing B_{t+1}. Denote then by $A_1^i,\ldots,A_{R_i}^i$ $(i = 1,\ldots,\ell)$.

Let $K_{t+1} = \{A_J^i : A_J^i \not\subset A,$ if $A \in F_t,\ 1 \leq J \leq R_i,\ 1 \leq i \leq \ell\} \cup F_t$.

Theorem 3.1. ([5]) For every t $(1 \leq t \leq m)$ then $K_t = \{B_1,\ldots,B_t\}^{-1}$. That is $K^{-1} = K_m$.

It can be seen that K and K^{-1} are determined uniquely by each other. Because of this fact, the determination of K^{-1} does not depend on the order of sequence $\{B_1,\ldots,B_m\}$. From the theorem 3.1 we construct a following algorithm.

Algorithm 3.2: find all the anti keys.

INPUT: $K = \{B_1,\ldots,B_m\}$ is a Sperner-system over $\Omega = \{1,\ldots,n\}$.

OUTPUT: M, where $M = K^{-1}$

DATA STRUCTURES:

1) K[1:m] is an array, which contains the elements of K.
2) M[1:p] is an array, which contains the elements of K^{-1} and $p = \binom{n}{\lfloor n/2 \rfloor}$.
3) N[1:p] is an array, which contains A_J^i-s.
4) A1,D1,D2,D3,D4 are work variables.

```
Procedure  ANTIKEYS  (K,M):
           M[1] ← Ω;  D1 ← 1;

CIKLUS:    for   i ← 1   until  m  do
           begin  D2 ← 0;    D3 ← 0;
           for  J ← 1  until  D1  do
             begin if  K[i] ≤ M[J]  then for each  q ∈ K[i] do
                                 begin D2   D2+1;N(D2) ← M[J]\{q};
                                       go to UTOLSO1
                                       end;
               D3 ← D3+1; M(D3) ← M[J];

UTOLSO1: empty end; D4 ← D3;
         if  D3 = 0  then begin  D4 ← D2;  for  q ← 1
                                   until  D2  do
                                 M[q] ← N[q]; go to UTOLSO 3
                          end;
         for  q ← 1  until D2 do begin  A1 ← 1;
              for  p ← 1  until  D3  do
                   if  N[q] CM[p] then begin  A1 ← 0;
                                       go to USOLSO2 end;

UTOLSO2: if  A1 = 1  then begin  D4 ← D4+1; M(D4) ← N[q]  end
                                                    end;

UTOLSO3:  D1 ← D4
          end.
```

It is obvious that m is the number of iteration of CIKLUS. Denote $K_o = \{\Omega\}$.

If $K_i = \{X_1,\dots,X_{\tau_i}\} \cup F_i$, where $B_{i+1} \subseteq X_J$ $(1 \leq J \leq \tau_i)$ and $F_i = \{A \in K_i \quad B_{i+1} \not\subseteq A\}$, ℓ_i is the number of the elements of K_i, then in the $i+1$ -th iteration of CIKLUS the algorithm requires $O(n^2(\ell_i - \tau_i)\tau_i)$ elementary operations, where $\tau_o = \ell_o = 1$ if $\tau_i < \ell_i$, and the algorithm requires

$O(n^2(\ell_i\ \tau_i)$ elementary operations if $\ell_i = \tau_i$.

By Theorem 3.1, we have:

Corollary 3.3: The algorithm 3.2 determines the set of all anti keys and its time complexity is $O(n^2 \sum_{i=1}^{m-1} \tau_i t_i)$, where

$$t_i = \begin{cases} \ell_i - \tau_i & \text{if } \tau_i < \ell_i, \\ 1 & \text{if } \tau_i = \ell_i. \end{cases}$$

It can be seen that when there are only a few minimal keys (i.e. m is small). Algorithm 3.2 is very effective. In cases, for which $\ell_i \leq \ell_m$ $(\forall i : 1 \leq i \leq m-1)$, it is obvious that our algorithm requires a number of elementary operations which is not greater than $O(n^2|K|\ |K^{-1}|^2)$. Thus, in these cases Algorithm 3.2 finds K^{-1} in polynomial time in $|\Omega|$, $|K|$ and $|K^{-1}|$. In [7], it has been proved that the worst-case time of our algorithm can not be than exponential in the number of attributes.

Example 3.4. Let $\Omega = \{1,2,3,4,5,6\}$ and $K = \{(1,2),(2,3,4),(2,4,5),(4,6)\}$.

From Theorem 3.1 and Algorithm 3.2 we have

$K_1 = \{(1,3,4,5,6),(2,3,4,5,6)\}$;

$K_2 = \{(1,3,4,5,6),(2,3,5,6),(2,4,5,6)\}$;

$K_3 = \{(1,3,4,5,6),(2,3,5,6),(2,4,6)\}$;

$K_4 = \{(2,3,5,6),(1,3,4,5),(1,3,5,6),(2,4)\}$

It is obvious that $K^{-1} = K_4$.

We consider the following matrix:

The attributes:

$$M = \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \left(\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 2 \\ 0 & 3 & 0 & 3 & 0 & 0 \\ 4 & 0 & 4 & 0 & 4 & 4 \end{array} \right) \end{array}$$

By [4], M represents K.

Now, we construct an algorithm, which find the set of all minimal keys. In this paper, it is last algorithm. In first time we have

Theorem 3.5: Let H be a Sperner-system over Ω and $H^{-1} = \{B_1,\ldots,B_m\}$ is the set of anti keys of H, $K \subseteq H$. Then: $K \subset H$ and $K \neq \emptyset$ if and only if there is a $B(B \in P(\Omega))$ such that $B \in K^{-1}$ and $B \not\subset B_i$ $(\forall i : 1 \leq i \leq m)$.

Proof. Suppose that there exists a B such that $B \in K^{-1}$ and $B \not\subseteq B_i$ $(\forall i : 1 \leq i \leq m)$. From the definition of the set of anti keys and by $K^{-1} \neq \emptyset$. We have $K \neq \emptyset$, and for all $C(C \in K)$, B not contains C. If there is a B_i such that $B_i \in H^{-1}$ and $B_i \subset B$, then it is obvious that B is a key. If $H^{-1} \cup B$ is a Sperner-system, then by Theorem 1.1 there exists a closure operation F such that $H = K_F$. It is clear that if $F(B) \neq \Omega$, then from Lemma 1.2 there is a $B_i (B_i \in H^{-1})$ such that $F(B) \subseteq B_i$. Consequently, $B \subseteq B_i$. This conflicts with the fact that $B \not\subseteq B_i$ $(\forall i : 1 \leq i \leq m)$.

That is B is a key. Hence there is an A ($A \subseteq \Omega$) such that A B and $A \in H \setminus K$.
It is easy to see that $K \subset H$.

Conversely, we suppose that $K \subset H$ and $K \neq \emptyset$. It is obvious that there is an A such that $A \in H \setminus K$. From H is a Sperner-system we have $A \cup K$ is a Sperner-system. Denote B the biggest set such that $A \subseteq B$ and $B \cup K$ is also a Sperner-system. It is clear that, B always exists and from the definition of anti keys we have $B \in K^{-1}$.
It can be seen that $A \not\subseteq B_i$ $(\forall i : 1 \leq i \leq m)$ by $A \in H$ and from Lemma 1.4. By A B we have $B \not\subseteq B_i$ $(\forall i : 1 \leq i \leq m)$.
The theorem is proved.

Let $K = \{B_1,\ldots,B_m\}$ be a Sperner-system over Ω.
We have to construct H, where $H^{-1} = K$. We construct H by induction.

Step i+1: If there is a $B \in K_i^{-1}$ such that
$B \not\subseteq B_J$ $(\forall J : 1 \leq J \leq m)$, then by an algorithm which find a minimal key, we construct one minimal key, A_{i+1} $(A_{i+1} \subseteq B)$ and let $K_{i+1} = K_i \cup \{A_{i+1}\}$.
Conversely, let $H = K_i$.

Corollary 3.6: there exists a natural number p such that $K_p = H$.

Not, we construct a following algorithm by the aid of the corollary 3.6.

Algorithm 3.7: find the set of all minimal keys.

INPUT: $K = \{B_1,\ldots,B_m\}$ is a Sperner-system over
$\Omega = \{1,\ldots,n\}$.

OUTPUT: M, where $M^{-1} = K$. (That is M is the set of all minimal keys).

DATA STRUCTURES:

1) K[1:m] is an array, which constains the elements of K.
2) M[1:p] is an array, which constains the set of all minimal keys and $p = \binom{n}{\lfloor n/2 \rfloor}$
3) L[1:p], N[1:p] are workarrays.
4) B, KEY, A1, D1, D2, D3, D4, D5, D6 are work variables.
5) MINIKEY5 (K,B) is a procedure, which find one minimal key.

Procedure MINIKEY SET (K,M):

```
D5  1; select ELEM from (Ω\K[1]); B ← K[1] ∪ {ELEM};
M[1] ← MINIKEY5(K,B); KEY ← M[1];
L[1] ← Ω; D1 ← 1;  D2 ← 0;   D3 ← 0;

CIKLUS1:  for  i ← 1  until  D1  do
            begin  if  KEY ⊆ L[i]  then for each
                                        q ∈ KEY  do
                       begin D2 ← D2+1; N(D2)← L[i]\{a};
                             go to UTOLSO1
                       end;
            D3 ← D3+1; L(D3) ← L[i]

UTOLSO1:  empty end; D4 ← D3;
          if  D3=0  then begin  D4 ← D2; for  q ← 1
              until  D2  to  L[q] ← N[q]; go to UTOLSO3
                     end;
          for  q ← 1  until  D2  do begin  A1 ← 1;
               for  p ← 1   until  D3  do
                 if  N[q] ⊂ L[p] then begin  A1 ← 0; go to
                                             UTOLSO2 end;
```

```
UTOLSO2:  if  A1 = 1  then begin  D4 ← D4+1; L(D4) ← N[a] end
                                  end;

UTOLSO3:  D1 ← D4;

          D6 ← D1;

CIKLUS2:  for  i ← 1  until  D1  do
               for  p ← 1  until  m  do
                  if  L[i] ≠ K[p]  then begin
                                        KEY   MINIKEY5(K,L[i]),
                                  D5 ← D5+1; M(D5) ← KEY; D1 ← D6;
                                                  go to CIKLUS1
               end.
```

By the proof is analogous to thealgorithm 3.2 it is clear that if k isthe number of minimal keys (that is: k is the member of iterations of CIKLUS1-USOLSO3 section) then in the CIKLUS1-UTOLSO3 section the algorithm requires $O(n^2 \sum_{i=1}^{k-1} r_i t_i)$ elementary operations and in the CIKLUS2--section the algorithm requires $O(n \cdot m \sum_{i=1}^{k} \ell_i)$ elementary operations (see the algorithm 3.2).
Consequently, from Theorem 3.5 we have.

Corollary 3.8: The algorithm 3.7 determines the set of all minimal keys and its time complexity is

$$O(n(\sum_{i=1}^{k-1} (m\ell_i + n r_i t_i) + m^2)).$$

It is clear that the algorithm 3.7 is effective, if the number of minimal keys is not big. In [6], it has been proved that the worst-case time of algorithm 3.7 can not be more than exponential in the member of attributes.

Example 3.9: Let $\Omega = \{1,2,3,4,5,6\}$ and

Relation R:

0	1	0	1	0	0
1	0	1	0	0	1
2	1	0	2	3	1
3	2	0	1	0	1
1	1	0	1	3	0

It is clear that $M = \{(3,4,5),(2,3,4,6),(5,6),(1),(2,3,5)\}$, where $E_{14} = \{3,4,5\}$, $E_{15} = \{2,3,4,6\}$, $E_{24} = \{5,6\}$, $E_{25} = \{1\}$, $E_{35} = \{2,3,5\}$.

From Lemma 1.2, M is the set R.
By the aid of Algorithm 3.7 we construct the set K such that $K^{-1} = M$.

From the MINIKEY(M,(1,2)) we have $K_1 = \{A_1\} = \{1,2\}$ and $K_1^{-1} = \{(2,3,4,5,6),(1,3,4,5,6)\}$.

From the MINIKEY5(M,(2,3,4,5,6)) we have
$K_2 = \{A_1,A_2\} = \{(1,2),(4,5,6)\}$, and

$K_2^{-1} = \{(2,3,5,6),(2,3,4,6),(2,3,4,5),(1,3,5,6),(1,3,4,6),$
$(1,3,4,5)\}$.

and $K_2^{-1} = \{(2,3,5,6),(2,3,4,6),(2,3,4,5),(1,3,5,6),$
$(1,3,4,6),(1,3,4,5)\}$.

By the MINIKEY5(M,(2,3,5,6)) we have $A_3 = (3,5,6)$, and

$K_3 = \{A_1,A_2,A_3\}$, $K_3^{-1} = \{(2,5,6),(2,3,4,6),(2,3,4,5),$
$(1,5,6),(1,3,4,6),(1,3,4,5)\}$.

From the MINIKEY5(M,(2,5,6)) we have $A_4 = (2,5,6)$, and

$K_4^{-1} = \{(2,3,4,6),(2,3,4,5),(1,5,6),(1,3,4,6),(1,3,4,5)\}$.

According to the MINIKEY5(M,(2,3,4,5)), we have $A_5 = (2,4,5)$, $K_5 = \{A_1,A_2,A_3,A_4,A_5\}$ and

$K_5^{-1} = \{(2,3,4,5),(2,3,5),(1,5,6),(1,3,4,6),(1,3,4,5)\}$.

From the MINIKEY5(M,(1,5,6)) we have $A_6 = \{1,6\}$, and

$K_6^{-1} = \{(2,3,4,6),(2,3,5),(5,6),(1,3,4,5)\}$.

By the MINIKEY5(M,(1,3,4,5)), we have $A_2 = \{1,5\}$ and

$K_7^{-1} = \{(2,3,4,6),(2,3,5),(5,6),(3,4,5),(1,3,4)\}$.

It can be seen that $A_8 = (1,4)$ by the MINIKEY5(M,(1,3,4)).

Consequently, we have $A_9 = \{1,3\}$ and

$K_9^{-1} = \{(2,3,4,6),(2,3,5),(5,6),(3,4,5),(1)\}$.

That is:

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

is the set of minimal keys, by $K_9^{-1} = M$.

ACKNOVLEDGEMENT

The author would like to take this opportunity to express deep gratitude to *Professor Dr. Janos Demetrovics* for his help, valuable comments and suggestions.

REFERENCES

[1] W.W. Armstrong; Dependency Structures od Data Base Relationships, Importation Processing 75, North-Holland Publ. Co(1974) 580-583

[2] E.F. Codd; Relational model of data for large shared data banks. Communications of the ACM, 13, (1970) 377-384.

[3] J. Demetrovics; On the equivalence of candidate keys with Sperner-systems. Acta Cybernetica 4 (1979) 247-252.

[4] J. Demetrovics; Relációs adatmodell logikai és strukturális vizsgálata. MTA SZTAKI tanulmányok, Budapest, 114 /1980/.

[5] Vu Duc Thi; Remarks on closure operations. MTA SZTAKI Közlemßnyek, Budapest, 30(1984) 73-87.

[6] Vu Duc Thi; Relációs adatmodell antikulcsairól. Alkalmazott Matematikai Lapok /1986/ /to appear/.

[7] Vu Duc Thi; Minimal keys and anti keys. Acta Cybernetica (1986) (to appear).

Relációs adatbázisok minimális kulcsainak és anti-kulcsainak megkeresésére vonatkozó algoritmusok

Vu Duc Thi

Összefoglaló

A cikkben összesen hét algoritmus van. Ezekből öt egy minimális kulcs megkeresésére vonatkozik /kettő közülük lineáris/. Egy algoritmus az összes minimális kulcsot kiadja, egy pedig az összes anti-kulcsot.

АЛГОРИТМЫ ДЛЯ НАХОЖДЕНИЯ МИНИМАЛЬНЫХ КЛЮЧЕЙ И АНТИ-КЛЮЧЕЙ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

Ву Диц Тхи

Р е з ю м е

В статье рабработано семь алгоритмов. Пять из них касаются нахождения одного минимального ключа /два из них линейные/. Один алгоритм есть для нахождения всего множества минимальных ключей. Один для нахождения множества анти-ключей.

MTA SZTAKI Közlemények 33/1985 p.145-146

BOOK REVIEW

M. Vukobratović, N. Kirčanski, "Real-Time Dynamics of Manipulation Robots", Comm. and Control Eng. Series (Eds.: A. Fettweis, J.L. Massey, M. Thoma), Scientific Fundamentals of Robotics 4, Springer, Berlin-Heidelberg-New York-Tokyo, 1985, (Hard cover, with 43 figures), pp. 1-239.

In the book a new approach to the formation of robot dynamics is presented, to achieve the real-time model computation using up-to-date microcomputers. The generation of the nonlinear dynamic robot model in analytical form based on new theoretical results is described in the book. This give new possibilities concerning real-time applications. A closed-form algorithm is developed for dynamic model construction, which is less numerically burdened than previous algorithms. The linearized robot model in analytical form is also developed. The algorithm for linearized model construction may be posed either on the closed-form model or on a specially defined operator for partial derivation of polynomial matrices. The sensitivity and approximate models in numeric-symbolic domain are also developed.
In the last chapter examples of various industrial robots are presented.

The book is intendent for engineers engaged in applied robotics, especially in studying dynamics of robotic systems, synthesizing control algorithms and, first of all, in their microcomputer implementation for actual, complex tasks arising in industrial robotics. It is also intended for students enrolled in post-graduate robotics courses.
The book is divided into 5 chapters: Ch. 1. Survey of computer-aided robot modelling methods; Ch. 2. Computer-aided method for closed-form dynamic robot model construction; Ch. 3. Computer-aided generation of numeric-symbolic robot model; Ch. 4. Model optimization and real-time program-code generation; Ch. 5. Examples.

Jürgen Ackermann, "Sampled-Data Control Systems. (Analysis and Synthesis, Robust System Design)", Comm. and Control Eng. Series (Eds.: A. Fettweis, J.L. Massey, M. Thoma), Springer, Berlin-Heidelberg-New York-Tokyo, 1985, (Hard cover, with 152 figures), pp. 1-596.

The book is a revised translation of the second German edition of 1983 (the first German edition appeared in 1972). While the first edition covered the analysis and synthesis of sampled-data systems, the second one extended the scope to design, in particular design for robustness of control system properties with respect to uncertainty of plant parameters. This book provides the fundamental theory for the analysis and synthesis of the resulting sampled-data control systems. It introduces to the design of controllers for robustness against structured plant uncertainties. It paves the way for interactive computer graphics design in parameter spaces. The hardware implementation of digital controllers for continuous plants is provided by microprocessors. The book is intended for engineers in industry as well as for graduate students and teachers of control engineering courses. It may be used in university courses at different levels.

The book has 9 chapters and 4 appendices:
Ch. 1. Introduction; Ch. 2. Continuous systems;
Ch. 3. Modelling and analysis of sampled-data systems;
Ch. 4. Controllability, choice of sampling period and pole assignment; Ch. 5. Observability and observers;
Ch. 6. Control loop synthesis; Ch. 7. Geometric stability investigation and pole region assignment; Ch. 8. Design of robust control systems; Ch. 9. Multivariable systems;
App. A. Canonical forms and further results from matrix theory; App. B. The z-transform; App. C. Stability criteria;
App. D. Application examples.