KYBERNETIKA — VOLUME 22 (1986), NUMBER 2

EQUITRAN: A COMPUTER PROGRAM FOR ANALYSIS OF A NONDETERMINISTIC DISCRETE DYNAMIC SYSTEM

PETR KŮRKA

A nondeterministic discrete dynamic system determines a transition graph, whose vertices are states of the system and whose oriented edges are possible transitions between states. We present here a computer program, which creates such a transition graph on the base of given transition rules and analyses its structural properties.

1. INTRODUCTION

Discrete dynamic systems offer a convenient tool for modelling complex dynamic systems encountered in biology and behavioural sciences. Systems of this kind are usually described by qualitative rather than numerical variables which specify whether specified substances are synthesized, whether some kind of behavioural pattern is going on, etc. The dynamics of the system is expressed by rules specifying under what conditions changes in the values of these variables happen. When such a system is formalized, we obtain a finite number of states (characterized by the values of the variables) and a set of rules which determine possible transitions between these states. Usually there are several transitions leading out of a given state, so the system is nondeterministic. The system stays in a state for an indefinite time (there are no assumptions on the probabilistic distribution of this time) and then switches to some other state, to which a transition leads (again we do not consider the probabilities of these switchings).

The trajectories of the system may be viewed therefore as paths in its transition graph, whose vertices are states and whose oriented edges are possible transitions. When this transition graph is too large to be manageable, automatic methods are required to find its most important structural properties determining the behaviour of the system. Of interest are for example ergodic states, ergodic sets of states or cycles of states, which determine possible ultimate destination of the system. A general method for achieving this goal is to define an equivalence on the set of states and

construct the factor graph, whose vertices are equivalence classes, and there is a transition from a set A to B iff there are states $x \in A$, $y \in B$ and a transition from x to y in the original graph. Thus equivalence classes are the "macrostates", whose inner structure is not investigated and the dynamics of the system is viewed, with substantial simplification, as a flow through macrostates. The equivalence may be either forced on the graph from outside or it may be determined by the structure of the graph alone.

This method is implemented in a computer program EQUITRAN (EQUIvalences on TRANsition graphs), which reads the table of transition rules of a discrete dynamic system, creates its transition graph and factorizes it through various equivalences as specified by the user.

2. NONDETERMINISTIC DISCRETE DYNAMIC SYSTEMS

Let us review here some methods of defining a nondeterministic discrete dynamic system and its transition graph.

2a. Transitional system

The system is described by n discrete variables assuming values in finite sets $W_1, \ldots, W_n$. A state of the system is a vector $Y = (Y_1, \ldots, Y_n)$ where $Y_i \in W_i$ for $1 \leqq i \leqq n$. The dynamics of the system is determined by a finite set of transition rules. Suppose $\times$ is a special symbol not contained in any W_i. A transition rule is a pair of vectors $U \to V$, where $U = (U_1, \ldots, U_n)$, $V = (V_1, \ldots, V_n)$ $U_i, V_i \in W_i \cup \{\times\}$. We say that there is a transition from a state Y to a state Z when $Y \neq Z$ and there exists a transition rule $U \to V$ such that

$$(1) \qquad (\forall i \leqq n)\,(U_i = \times \text{ or } U_i = Y_i)$$

$$(2) \qquad (\forall i \leqq n)\,((V_i = \times \text{ and } Z_i = Y_i) \text{ or } Z_i = V_i)\,.$$

Thus the left-hand side U of a transition rule specifies to which states the rule may be applied. The cross $\times$ in U means "no matter what is the value of corresponding variable". The right-hand side V specifies the new values of the variables. The cross here means "no change of the value of the corresponding variable". A transition rule expresses the idea that for specified values of some variables other variables assume new prescribed values. The transitional system is a generalization of kinetic logic (see [5]) to many-valued variables and simultaneous nondeterministic changes of several variables.

2b. Kinetic logic

The system is described by n boolean variables. The dynamics of the system is described by n n-ary boolean functions $f_1, \ldots, f_n$: $\{0, 1\}^n \to \{0, 1\}$. A state of the

system is a vector $Y = (Y_1, \ldots, Y_n)$, $Y_i \in \{0, 1\}$. There is a transition from a state Y to Z if

$$(3) \qquad Y \neq Z \quad \text{and} \quad (\forall i \leqq n)(Z_i = f_i(Y_1, \ldots, Y_n) \quad \text{or} \quad Z_i = Y_i).$$

Thus the functions f_i give trends of the variables i.e. values, which the variables tend to assume. In a transition from one state to another some variables change their values according to these trends.

A modified system of kinetic logic is obtained when we require that only one variable changes at a time. In this case we define that there is a transition form Y to Z if

$$(4) \qquad (\exists i \leqq n)(Z_i = f_i(Y_1, \ldots, Y_n) \neq Y_i \,\&\, (\forall j \neq i)(Z_j = Y_j)).$$

2c. Semi-Thue systems

Let V be a finite alphabet, V^* the set of words with letters from V. A state of the system is a word $y \in V^*$. The dynamics of the system is given by a finite set of productions of the form $u \to v$, where $u, v \in V^*$. There is a transition from a state y to z if there is a production $u \to v$ and words $p, q \in V^*$ such that $y = puq$, $z = pvq$. Contrary to the preceding system, the set of states is infinite here. Semi-Thue systems in connection with discrete dynamic systems are investigated for example in [3].

2d. Lindenmayer systems

Let V be a finite nonvoid alphabet. The states are again words from V^* and the dynamics is given by a finite set of productions of the form $x \to v$, where $x \in V$, $v \in V^*$. There is a transition from a state y to z when $y = y_1 y_2 \ldots y_n$, $y_i \in V$, $z = z_1 z_2 \ldots z_n$ for some $z_i \in V^*$, and $y_i \to z_i$, $i = 1, \ldots, n$ are productions of the system. Lindenmayer systems are treated for example in [4].

3. TRANSITION GRAPHS

A graph is a pair (X, E) where X is a finite set of vertices and $E \subseteq X \times X$ is an antireflexive relation, i.e. $x \in X \Rightarrow (x, x) \notin E$. We shall write $x \to y$ when $(x, y) \in E$. An equivalence on X is a reflexive, symmetric and transitive relation $\sim \subseteq X \times X$. It determines a partition $X/\sim$ consisting of equivalence classes of $\sim$. An equivalence $\sim$ is finer than $\approx$ ($\approx$ is coarser than $\sim$) if $x \sim y$ implies $x \approx y$. If (X, E) is a graph and $\sim$ is an equivalence on X, then the factor graph $(X, E)/\sim$ is a graph whose vertices are equivalence classes of $\sim$ and there is an edge $A \to B$ for $A, B \in X/\sim$ if $A \neq B$ and there exists an edge $x \to y$ in the original graph for some $x \in A$, $y \in B$. A path in a graph (X, E) is a sequence of vertices $x_1 \to x_2 \to \ldots \to x_n$. The factorization of this path is the path $(A_i)_{i=1,\ldots,m}$ in $(X, E)/\sim$ to which there exist indices

$1 = i_1 < i_2 < \ldots < i_{m+1} = n + 1$ such that $x_k \in A_j$ for $i_j \leqq k < i_{j+1}$. The factorization of a path is therefore its image in the graph.

The factor graph should retain essential dynamic properties of the system. A path in the factor graph, for example, should correspond to some possible trajectory of the system, i.e. it should be a factorization of some path in the original graph. This is not true in general, so this condition may serve as a criterion of acceptability of an equivalence.

Definition 1. Let (X, E) be a graph and $\sim$ an equivalence on X. We say that $\sim$ reflects paths, if any path in $(X, E)/\sim$ is a factorization of some path in (X, E).

3a. Destination division of an equivalence

Suppose we have a transitional system with variables $Y_1, \ldots, Y_n$ and we are interested only in time development of specified variables $(Y_i)_{i \in M}$. We can define an equivalence $\sim$ by

$$Y \sim Z \quad \text{iff} \quad Y_i = Z_i \quad \text{for all} \quad i \in M . \tag{5}$$

This equivalence, however, need not reflect paths in the sense of Definition 1, i.e. it identifies states from which the system may evolve in quite different ways. A finer equivalence is therefore required. A possible solution to this problem is the following

Proposition 1. Let $\approx$ be an equivalence on a graph (X, E). Then there exists the coarsest equivalence $\sim$ which is finer than $\approx$ and satisfies

$$(\forall x, y)\,(x \sim y \Rightarrow (\forall x')\,(x \to x' \Rightarrow (\exists y')\,(y \to y' \,\&\, y' \sim x'))) \tag{6}$$

We say that $\sim$ is the destination division of $\approx$.

Proof. Construct nondeterministically a sequence of partitions $\mathscr{A}_0, \mathscr{A}_1, \ldots, \mathscr{A}_p$ as follows. $\mathscr{A}_0$ is $X/\approx$. Suppose that $\mathscr{A}_i$ has been constructed and let there exist $A,\ B \in \mathscr{A}_i$ such that for the set $C = \{x \in A \mid (\exists x' \in B)\,(x \to x')\}$ neither $C = \emptyset$ nor $C = A$. Define then next partition $\mathscr{A}_{i+1} = \mathscr{A}_i - \{A\} \cup \{A - C, C\}$. This procedure ends when for all sets $A, B \in \mathscr{A}_p$ either $C = \emptyset$ or $C = A$. Define $X/\sim$ as $\mathscr{A}_p$. Clearly $\sim$ is finer than $\approx$ and satisfies (6). Suppose $\sim'$ is another equivalence which is finer than $\approx$ and satisfies (6) but is not finer than $\sim$. Then for some $i < p$ $\sim'$ is finer than $\mathscr{A}_i$ but not finer than $\mathscr{A}_{i+1}$. Since $\mathscr{A}_{i+1} = \mathscr{A}_i - \{A\} \cup \{A - C, C\}$ there exist $x \in C,\ y \in A - C$ with $x \sim' y$. Furthermore there exists $x' \in B$ with $x \to x'$ and by (6) there exists $y' \sim' x'$ such that $y \to y'$. Since $\sim'$ is finer than $\mathscr{A}_i$, $y' \in B$ and this is a contradiction. □

Note. The condition (6) in Proposition 1 may be replaced by a less strict condition

$$(\forall x, y)\,(x \sim y \Rightarrow (\forall x')\,(x \to x' \,\&\, x' \neq x \Rightarrow (\exists y')\,(y \to y' \,\&\, x' \sim y')) \tag{7}$$

and Proposition 1 remains valid.

Proposition 2. The destination division of an equivalence reflects paths.

Proof. Let $(A_i)_{i=1,\ldots,n}$ be a path in $(X, E)/\sim$. Choose some $x_1 \in A_1$ and suppose that we have constructed a path $(x_i)_{i=1,\ldots,k}$, $k < n$, whose factorization is $(A_i)_{i=1,\ldots,k}$. There exist $x \in A_k$, $y \in A_{k+1}$ with $x \to y$. By Proposition 1 (either condition 6 or 7) there exists $x_{n+1} \in A_{k+1}$ with $x_n \to x_{n+1}$.

3b. The communication equivalence

Definition 2. Let (X, E) be a graph. Two states $x, y \in X$ communicate if either $x = y$ or there exists a path from x to y and a path from y to x.

The equivalence classes of communication equivalence are also called strongly connected components (see [1]). A communication set from which no arrow leads in the factor graph is called ergodic set. Ergodic sets represent possible ultimate destination of the dynamic system. The system may enter them but can never leave them.

Proposition 3. The communication equivalence reflects paths and it is the finest equivalence whose factor graph does not contain cycles.

The proof of this proposition is trivial.

3. The destination equivalence

Proposition 4. Let (X, E) be a graph. There exists the finest equivalence $\sim$ (destination equivalence) satisfying the condition

$$(8) \qquad (\forall x, y)\,((\forall x')\,(x \to x' \Rightarrow (\exists y')\,(y \to y' \,\&\, y' \sim x')) \,\&\, (\forall y')\,(y \to y' \Rightarrow \Rightarrow (\exists x')\,(x \to x' \,\&\, y' \sim x')) \Rightarrow x \sim y)$$

Proof. Consider a sequence of partitions $\mathscr{A}_0, \mathscr{A}_1, \ldots, \mathscr{A}_p$ where $\mathscr{A}_0$ is identity equivalence, constructed recursively as follows: If there exist $A, B \in \mathscr{A}_i$ $A \neq B$ such that for all $x \in A$, $y \in B$

$$(\forall x')\,(x \to x' \Rightarrow (\exists y')\,(y \to y' \,\&\, y' \sim_i x')) \,\&$$
$$(\forall y')\,(y \to y' \Rightarrow (\exists x')\,(x \to x' \,\&\, y' \sim_i x'))$$

then let $\mathscr{A}_{i+1} = \mathscr{A}_i - \{A, B\} \cup \{A \cup B\}$. (Here $\sim_i$ is the equivalence corresponding to the partition $\mathscr{A}_i$). Let $\mathscr{A}_p$ be the first partition, to which this procedure cannot be applied. It is easy to see that this is the required equivalence. □

Proposition 5. The destination division of the total equivalence $X \times X$ is coarser than the destination equivalence. Moreover, both these equivalences satisfy conditions (6) and (8).

Proof. Let $\mathscr{A}_0, \ldots, \mathscr{A}_p$ be the sequence of partitions from the proof of the preceding proposition. Clearly $\mathscr{A}_0$ satisfies (6). Let $\mathscr{A}_i$ satisfy (6) and let $A, B \in \mathscr{A}_i$ be such

that $\mathscr{A}_{i+1} = \mathscr{A}_i - \{A, B\} \cup \{A \cup B\}$. If $x \in A$, $y \in B$ then $(\forall x')(x \to x' \Rightarrow ((\exists y') . . (y \to y' \,\&\, y' \sim_i x'))$ by the proof of the preceding proposition. It follows that $\mathscr{A}_{i+1}$ satisfies (6); consequently the destination equivalence satisfies both conditions (6) and (8). Analogously it may be proved that the destination division of the total equivalence satisfies (8). Since by Proposition 4 the destination equivalence is the finest equivalence satisfying (8). It must be finer than the destination division of the total equivalence. □

Counterexample. Let (X, E) be a cycle, i.e. $X = \{x_1, \ldots, x_n\}$. $E = \{(x_1, x_2), (x_2, x_3), \ldots, (x_n, x_1)\}$. Then the destination equivalence is the identity, while the destination division of total equivalence is the total equivalence. This proves that these two equivalences need not be equal. They are equal, however, when the graph does not contain cycles.

Proposition 6. Let (X, E) be a graph not containing cycles. Then the destination division of total equivalence and the destination equivalence coincide.

Proof. Define a partition $\{X_0, \ldots, X_p\}$ as follows: $X_0 = \{x \in X \mid (\exists y)(x \to y)\}$ $X_{i+1} = \{x \in X - (X_0 \cup \ldots \cup X_i) \mid (\forall y)(x \to y \Rightarrow y \in X_0 \cup \ldots \cup X_i)$. Here p is the first number for which $X_{p+1} = \emptyset$. Define by induction equivalences $\sim_i$ on X_i by the formula

$$x \sim_i y \quad (\forall x')(x \to x' \Rightarrow (\exists y')(y \to y' \,\&\, x' \sim_j y' \quad \text{for some} \quad j < i)$$
$$\&\, (\forall y')(y \to y' \Rightarrow (\exists x')(x \to x' \,\&\, x' \sim_j y' \quad \text{for some} \quad j < i) .$$

Let the equivalence $\sim$ on X be the union of all $\sim_i$, and let us prove that it is both destination division of total equivalence and destination equivalence. It is easy to see that $\sim$ satisfies (6). Let $\approx$ be another equivalence satisfying (6). Then we can prove by induction that $x \in X_i$, $y \approx x \Rightarrow y \in X_i$, $y \sim_i x$, consequently $\approx$ is finer than $\sim$, so $\sim$ is the destination division of total equivalence. Conversely we can prove by induction that if $x \in X_i$ and $(\forall x')(x \to x' \Rightarrow (\exists y')(y \to y' \,\&\, y' \sim x')) \,\&$ $(\forall y')(y \to y' \Rightarrow (\exists x')(x \to x' \,\&\, y' \sim x'))$ then $y \in X_i$ and $y \sim x$, so $\sim$ satisfies (8). Finally we can prove that if $\approx$ is an equivalence satisfying (8) and $x \sim_i y$, then $x \approx y$, so $\approx$ is coarser than $\sim$, and $\sim$ is the destination equivalence. □

Proposition 7. The destination equivalence reflects paths.

The proof follows from the fact that the destination equivalence satisfies condition (6).

4. THE PROGRAM EQUITRAN

For the purposes of analysis of discrete dynamic systems we developed a computer program, which is based on the transitional system described in Section 2a. The syntax of input data is summarized in Fig. 1. The parameter NVAR determines

the number of variables. For Ith variable DIM(I) determines the set of its values, which is $(\{0, 1, \ldots \text{DIM}(I) - 1\}$. A state of the system is then any integer vector $Y = (Y(1), \ldots Y(\text{NVAR}))$ where $0 \leqq Y(I) < \text{DIM}(I)$. Initial states are given by a vector INI = (INI (1), ... INI (NVAR)), where INI $(I) \in \{\times, 0, 1, \ldots \text{DIM}(I) - 1\}$.

```
EQUITRAN INPUT DATA (DATA SET SYSIN)
 1.FREE FORMAT ASSIGNMENTS TO PARAMETERS
   NVAR,TITLE,NROW,NCOL,NARY,SETS,STATES,CYCLES.
   ONLY NVAR ASSIGNMENT IS MANDATORY.
   THE ORDER OF ASSIGNMENTS IS ARBITRARY.
   THESE ASSIGNMENTS END WITH SEMICOLON.
 2.FIXED FORMAT DATA FOR PARAMETERS
   NAM,DIM,INI,USE,L,R.
   THEY BEGIN IN FIRST CARD AFTER SEMICOLON.
```

CARD	COLUMN	NAME	VALUES	DEFAULT	INTERPRETATION
		NVAR	[illegible]		NUMBER OF [illegible]
		TITLE	[illegible]	'ELET'	TITLE OF [illegible]
		NROW	[illegible]	60	NUMBER OF ROWS PER PAGE
		NCOL	[illegible]	120	NUMBER OF COLUMNS PER PAGE
		NARY	[illegible]	'NO'	PRINT TRANSITION GRAPH
		SETS	[illegible]	'COM'	TYPE OF SETS
			'COMPUTATIONAL',		(FIRST THREE CHARACTERS
			'RELATIONAL')		ARE SIGNIFICANT ONLY.)
		STATES	('YES','NO')	'YES'	PRINT STATES IN TRAN. GRAPH
		CYCLES	('YES','NO')	'NO'	PRINT CYCLES
			;		ENDS FREE FORMAT PARAMETERS
1	5*I-4	NAM(I)	CHARACTER(4)		NAME OF I-TH VARIABLE
2	I	DIM(I)	INTEGER<=9		DIMENSION OF I-TH VARIABLE
3	I	INI(I)	(X,0,..DIM(I)-1)		INITIAL VALUE OF I-TH VAR.
4	I	USE(I)	(Y,N) (YES,NO)		USE OF I-TH EQUIVALENCE
4+K	I	L(K,I)	(X,0,..DIM(I)-1)		I-TH COMPONENT OF LEFT SIDE
					OF K-TH TRANSITION RULE
4+K	I+NVAR	R(K,I)	(X,0,..DIM(I)-1)		I-TH COMPONENT OF RIGHT SIDE
	+2				OF K-TH TRANSITION RULE
4+K	2*NVAR	COM(K)	CHAR(78-2*NVAR)		COMMENT OF K-TH RULE
	+3				

Fig. 1.

Here the cross $\times$ means that states with any value of the variable in question are initial; we say that a state Y is initial if

$$(\forall I)\,(Y(I) = \text{INI}(I) \quad \text{or} \quad \text{INI}(I) = \times)\,. \tag{10}$$

Transition rules are given by pair of vectors $(L(K, 1), \ldots L(K, \text{NVAR})) \to (R(K, 1), \ldots$ $\ldots R(K, \text{NVAR}))$, where $L(K, I)$, $R(K, I) \in \{\times, 0, 1, \ldots \text{DIM}(I) - 1\}$, $K = 1, \ldots$ $\ldots$ NRULES. Here $\times$ is again a special symbol; there is a transition from a state Y to Z iff there is K so that

$$(\forall I)\,(Y(I) = L(K, I) \quad \text{or} \quad L(K, I) = \times)\,, \quad \text{and} \tag{11}$$

$$(\forall I)\,(Z(I) = R(K, I) \quad \text{or} \quad (R(K, I) = \times \;\&\; Z(I) = Y(I))\,.$$

The vector USE specifies equivalence (5) from Section 3a by condition

$$Y \sim Z \quad \text{iff} \quad (\forall I)\,(\text{USE}(I) = Y \Rightarrow Y(I) = Z(I))\,. \tag{12}$$

The program first creates the transition graph of all states accessible from initial

ones along possible transitions, and if GRAPH = 'YES' has been specified, prints it out. When not all variables have been used, i.e. USE$(I) = N$ for at least one I, the destination division of the equivalence $\sim$ and its factor graph are determined. The original transition graph is in this case freed. The program then determines the factor graph of communication sets, and if SETS = 'DESTINATION' has been specified, its factorization through destination equivalence. The requested sets and their transition graph are then printed. Furthermore, the program can print all cycles of states.

The algorithm for determining the communication sets is taken from [1], the algorithm for determining the cycles of a communication set is adapted from [2]. The program is written in PL/1 (F compiler), and runs under OS/IBM operating system. It works well with transition graphs of several hundred states.

5. AN EXAMPLE: GENETIC REGULATORY SYSTEM

A typical nondeterministic discrete dynamic system is a genetic regulatory network (see [5] pp. 217–225), which consists of operons, genes and regulatory sites of DNA molecule. An example of such network can be seen in Fig. 2. Here we have

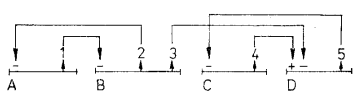

Fig. 2.

an operon A, which contains a gene for protein 1, operon B coding proteins 2 and 3, operon C coding protein 4 and operon D coding protein 5. Furthermore, these operons contain positive or negative regulatory sites to which specified proteins bind. When a protein binds to a negative regulatory site, the operon is inactivated and its proteins are not synthesized. On the other hand, the operon which contains a positive regulatory site may be active only if regulatory protein binds to this site. The question is which operons will be induced and which will be repressed.

The state of such a system may be described by four boolean variables A, B, C, D, which assume value 1 when operon is induced, i.e. its proteins are synthesized, and 0 otherwise. The transition rules for this system are given in the table in Fig. 3. For example if $B = 0$ then the protein 2 is not synthesized, so the operon A may be activated to $A = 1$. Thus we have the first transition rule $\times 0 \times \times \rightarrow 1 \times \times \times$ (in the printout the crosses are replaced by spaces). Since the scheme in Fig. 2 does not specify the concurrence of events, there are several possibilities how to express it with a set of transition rules. For example the second rule may be split into two independent rules $B = 1 \Rightarrow A = 0$ and $B = 1 \Rightarrow D = 0$ ($\times 1 \times \times \rightarrow 0 \times \times \times$ and $\times 1 \times \times \rightarrow$

$\rightarrow \times \times \times 0$) and the order of the changes ($A = 0$, $D = 0$) would be indeterminate. Conversely, rules 1 and 7 may be combined into two other rules $B = 0$, $C = 0 \Rightarrow$ $\Rightarrow A = 1$ and $B = 0$, $C = 1 \Rightarrow A = 1$, $D = 1$ and the indeterminacy of the system would be reduced.

```
EEEEE   QQQ   U   U  I  TTTTT  RRRR     A    N   N
E      Q   Q  U   U  I    T    R   R   A A   NN  N
EEE    Q  QQ  U   U  I    T    R   R  A   A  N N N
E      Q   Q  U   U  I    T    RRRR   AAAAA  N  NN
EEEEE   QQQ Q  UUU   I    T    R   R  A   A  N   N

INPUT DATA:
TITLE:                          GENETIC REGULATORY SYSTEM
NUMBER OF VARIABLES:              4
FORMAT OF PRINTED PAGE:          80 COLUMNS X 60 ROWS
PRINT TRANSITION GRAPH:         YES
PRINT ...:                      COMMUNICATION
PRINT STATES:                   YES
PRINT CYCLES:                   YES

VARIABLE NAMES:
                        ABCD|ABCD
                            |
                            |
                            |
NUMBER OF VALUES:       2222
INITIAL VALUES          X000
... ... ...             YYYY
TRANSITION RULES:
                         0  |1              A = ¬B
                         1  |0  0
                        0   | 1             B = ¬A
                        1   | 0
                           0|  1            C = ¬D
                           1|  0
                         01 |   1           D = ¬B & C
                          0 |   0

THE TRANSITION GRAPH HAS BEEN CREATED
NUMBER OF STATES:    12
NUMBER OF EDGES:     20

GENETIC REGULATORY SYSTEM          TRANSITION GRAPH OF STATES
STATE  DESCENDANT
...    ...  ...  ...
0001    0000 0101 1001
0010    0011 0110 1010
0011    0001 0111 1011
0100    0110
0101    0100
0110
0111    0101 0110
1000    1010
1001    1000
1010    1011
1011    1001

THE TRANSITION GRAPH HAS BEEN PRINTED
```

Fig. 3.

The table of transition rules is the main part of the input of the program. Other necessary data are number of variables (in our case NVAR = 4), number of values of each variable (here 2 – binary variables), initial value of each variable (here $A = \times$, which means indeterminacy and $B, C, D = 0$), and which variables are to be used (here all variables are used).

The input data together with default values are summarized on the first page

of the computer printout (Fig. 3), which also gives all possible transitions between states accessible frim the initial ones. On the next page (Fig. 4) the transition graph of communication sets is printed. A set is displayed by its reference number and by its summary state. In a summary state there are values of variables which do not

```
GENETIC REGULATORY SYSTEM          TRANSITION GRAPH OF COMMUNICATION SET
SET       DESCENDANTS
  1:00XX   4:0100   6:0110   2:0111   5:10XX   3:0101
  2:0111   6:0110   3:0101
  3:0101   4:0100
  4:0100   6:0110
  5:10XX
  6:0110

THE TRANSITION GRAPH OF COMMUNICATION SETS HAS BEEN PRINTED


GENETIC REGULATORY SYSTEM          COMMUNICATION SETS

  1
------
0000
0010
0011
0001

  2
------
0111

  3
------
0101

  4
------
0100

  5      6
------  ------
1000   0110
1010
1011
1001

COMMUNICATION SETS HAVE BEEN PRINTED


GENETIC REGULATORY SYSTEM          CYCLES OF SET    1
      0000->0010->0011->0001->0000

GENETIC REGULATORY SYSTEM          CYCLES OF SET    5
      1000->1010->1011->1001->1000

THE CYCLES OF COMMUNICATION SETS HAVE BEEN PRINTED

PROGRAM EQUITRAN TERMINATED NORMALLY
```

Fig. 4.

vary throughout the set and $\times$'s for variables which vary throughout the set. Next, all communication sets are listed. This print is organized so that the possible transitions lead only downwards. In the last row there are ergodic sets 5, 6, from which no exist is possible. Finally all cycles of states are printed. Here both sets 1 and 5 are formed by one cycle.

We see from this analysis that the system may proceed through the cycle of states

$0000 \to 0010 \to 0011 \to 0001 \to 0000$. This cycle is, however, unstable. Once the system leaves it, it enters after some delay either a stable state 0110 or a stable cycle $1000 \to 1010 \to 1011 \to 1001 \to 1000$.

```
GENETIC REGULATORY SYSTEM          TRANSITION GRAPH OF REDUCED        SET
SET        DESCENDANTS
  1:00XX   3:10XX   2:01XX
  2:01XX
  3:10XX

THE TRANSITION GRAPH OF REDUCED        SETS HAS BEEN PRINTED


GENETIC REGULATORY SYSTEM          REDUCED        SETS

  1     2     3
-----  ----  ----
0001   0101  1001
0011   0111  1011
0010   0110  1010
0000   0100  1000

REDUCED        SETS HAVE BEEN PRINTED


GENETIC REGULATORY SYSTEM          ERGODIC        SETS

  1     2
---   ---
01    10

ERGODIC SETS HAVE BEEN PRINTED
PROGRAM EQUITRAN TERMINATED NORMALLY
```

Fig. 5.

Suppose now that we are interested only in behaviour of variables A and B, so the vector of used variables would be YYNN. The results of this run ate in Fig. 5 (first part of the printout has been omitted). The states are now grouped into reduced sets consisting of states, which agree in the values of used variables. The printout consists of transition graph of reduced sets, list of reduced sets and list of ergodic sets. We see that as long as only A and B variables are considered, the system has two possible stable states 01 and 10.

(Received July 4, 1984.)

REFERENCES

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman: The Design and Analysis of Computer Algorithms. Addison-Wesley, Massachusetts, 1976.

[2] A. T. Berztiss: Data Structures — Theory and Practice. Academic Press, New York 1975.

[3] P. Kůrka: Ergodic languages. Theoret. Comput. Sci. *21* (1982), 351—355.

[4] A. Lindenmayer and G. Rozenberg (eds.): Automata, Languages, Development. North Holland, Amsterdam 1976.

[5] R. Thomas (ed.): Kinetic Logic, a Boolean Approach to the Analysis of Complex Regulatory Systems. (Lecture Notes in Biomathematics 29.) Springer-Verlag, Berlin 1979.

[6] R. Thomas: Logical description, analysis and synthesis of biological and other networks comprising feedback loops. Adv. Chem. Phys. *55* (1983), 247—282.

RNDr. Petr Kůrka, CSc., Oddělení biomatematiky Fyziologického ústavu ČSAV (Center of Biomathematics — Czechoslovak Academy of Sciences), Vídeňská 1083, 142 20 Praha 4. Czechoslovakia.