„COMPUTING" THE REFLECTIVE TRANSITIVE CLOSURE OF A FUZZY GRAPH: APPLICATIONS TO FUZZY RULE-BASED SYSTEM INTERPRETERS

KOSTADIN KRATCHANOV, IVAN STANEV

1. Introduction

The vast majority of today's knowledge-based systems are rule-based systems (RBS's). The development of their mathematical foundations is important and useful from both theoretical and practical point of view. An approach to grounding RPS's has been carried out in [1] - [6] et al.

On the other hand, some initial steps towards „fuzzification" of the mathematical notion of a RBS have also been made [7] - [9] thus contributing to the efforts of grounding and investigating the fuzzy RBS's. Some sort of fuzzy reasoning is included, in particular, in many commercial expert systems.

The one-step behaviour of a fuzzy RBS is a homogeneous (i.e. with coinciding source and target sets) fuzzy relation which is itself a composition of component fuzzy relations (or ordinary functions). The basic part of the (full, many-step) behaviour of the system is the reflective transitive closure (RTC) of the one-step behaviour.

In this report we investigate systematically, on a general level, non-recursive backtracking algorithms for computing such closure. This results in a methodology which can be used whenever developing universal or specialized RBS interpreters of any sort. Meanwhile we have to introduce a new mathematical notion – the RTC of a fuzzy graph. The backtracking algorithms we study are extentions of the traditional ones in the sense that they consider:

- relations which are fuzzy;
- relations which themselves are compositions of relations;
- relations with transit poles [10], [11].

The methodology has been used when designing the fuzzy interpreters algorithms of [8], as well as when drowing the very special algorithm of the interpreter for fuzzy robot control programs considered at the end of this report. The methodology comprises, of course, the case of non-fuzzy RBS interpreters (e.g. [3] - [6]). Questions parallel to those considered here (but for crisp relations only) have been discussed in [12]. Some inaccurances in the algorithms of [12] have been removed.

2. Preliminaries

The terminology and notations follow [5], [7], [8], [11], [13]. Some important preliminary notions are included in this sections.

A fuzzy relation from a set X to a set Y, $r: X \to Y$, is a function $r: X \times Y \to V_r$ where V_r is its valuation set. The fuzzy relation is homogeneous iff $X = Y$. The composition $s.\ r$: $X \to Z$ of fuzzy relations r: $X \to Y$ and s: $Y \to Z$ is given by

$$\langle x, z\rangle \to \bigoplus_{y \in Y} r(x, y) \square s(y, z)$$

where the operations $\oplus$ and $\square$ form a super-semiring [13]. In all the examples we use Zadeh's classical fuzzy logic in which $V_r = [0, 1]$ for any r, $\oplus = \max$, $\square = \min$. The pseudo-product, $r \times s$: $X_1 \times X_2 \to Y_1 \times Y_2$, of two fuzzy relations $r: X_1 \to Y_1$ and $s: X_2 \to Y_2$ is determined by

$$r \times s : \langle x_1, x_2, y_1, y_2 \rangle \mapsto r(x_1, y_1) \square s(x_2, y_2).$$

A fuzzy subset $\underline{X}$ of a set X is a function $\underline{X}: X \to V$. It can be equivalently considered as one of the fuzzy relations

$$\underline{X}^{(1)} : \underline{1} \to X : \langle 0, x \rangle \mapsto \underline{X}(x); \tag{1}$$

$$X^{(2)} : x \to \underline{1} : \langle x, 0 \rangle \mapsto \underline{X}(x); \tag{2}$$

$$\underline{X}^{(3)} : X \to X : \langle x, x' \rangle \mapsto \begin{cases} \underline{X}(x), \text{ if } x = x', \\ 0, \text{ otherwise.} \end{cases} \tag{3}$$

Evidently a fuzzy relation $X \to Y$ is a fuzzy subset of $X \times Y$, and vice versa.

A fuzzy graph is an ordered quadruple $G = \langle N, A, s, t\rangle$: where N is a set (of nodes), A is a fuzzy set (of arrows), s and t are mappings $A \to N$ determining the source and the target of each arrow respectively. Let V be the valuation set of the fuzzy set A. If the valuation degree of an arrow a in A is $v \in V$ then graphically we shall denote it

$$(n_1) \xrightarrow{a:v} (n_2) \quad \text{or simply} \quad (n_1) \xrightarrow{v} (n_2)$$

provided $s(a) = n_1$ and $t(a) = n_2$. Arrows whose valuation degrees are not greater than a fixed threshold $th \in V$ will not be included into the graphical representation of the graph. A fuzzy graph will be called simple iff

$$(\forall\, a, a' \in A)\ s(a) = s(a') \ \&\ t(a) = t(a') \Rightarrow a = a'.$$

Any homogeneous fuzzy relation $r: Y \to Y$ can be considered as a simple fuzzy graph $G_r = \langle Y, Y^2, s, t\rangle$ with $s(y_1, y_2) = y_1$ and $t(y_1, y_2) = y_2$. Conversely, each simple fuzzy graph $G = \langle N, A, s, t\rangle$ determines a homogeneous fuzzy relation r_G on N with

$$r_G(n_1, n_2) = v \Leftrightarrow (\exists\ a \in A)\ s(a) = n_1 \ \&\ t(a) = n_2 \ \&\ A(a) = v.$$

For the sake of simplicity we shall not distinguish between a homogeneous fuzzy relation, its corresponding simple graph, and the graphical representation of this graph. In this context a homogeneous fuzzy relation is a special case of a fuzzy graph.

Examples of homogeneous fuzzy relations are shown in Fig. 5a. (They correspond to simple fuzzy graphs.) Examples of fuzzy graphs are included in Fig. 5b.

A fuzzy graph $G=\langle N, A, s, t\rangle$ will be called adjusted if two fuzzy subsets, N_i and N_f, on N have been fixed. A node $n \in N$ is named initial, respectively final with valuation v iff $N_i(n)=v$, respectively $N_f(n)=v$. In the terminology of fuzzy relations N_i and N_f are conveniently represented by $N_i^{(1)}: \underline{1} \to N$, respectively $N_f^{(2)}: N \to \underline{1}$. Sometimes, however, more convenient is the representation

$$N \xrightarrow{N_i} N \xrightarrow{r_a} N \xrightarrow{N_f} N.$$

A triple of fuzzy relations,

$$\underline{1} \xrightarrow{r_i} Y \xrightarrow{r} Y \xrightarrow{r_f} \underline{1},$$

will be called an adjusted homogeneous fuzzy relation.

A fuzzy action on X is a fuzzy relation $X \to X$. A fuzzy condition (predicate) on X is a fuzzy subset of X. A fuzzy rule is an ordered pair, $[\underline{A}, \underline{C}]$, of a fuzzy subset of fuzzy actions and a fuzzy subset of fuzzy conditions.

A RBS is a universal computation model and has the general structure shown in Fig. 1. In contrast to the usual approach, the control unit in our RBS's is most powerful – it is a fuzzy recursive transition network or, as we call it, a recursive fuzzy automat. It can be represented (e.g. [8]) by an ordinary fuzzy

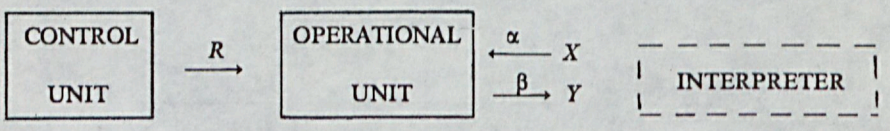


Fig. 1

automaton but with an infinite state set $Z=S\times T$ with $T=S^*$ playing the role of a stack. By D we denote the state set of the operational unit. R is a set of fuzzy rules on D. α and β are simple „coding" relations showing the interpretation of the data (these relations are not „computed" by the model). If $b: D\times Z \to D\times Z$ is the one-step behaviour of the system then the (many-step, overall) behaviour is by definition the composition of Fig. 2. Here Z_i and Z_f

$$X \xrightarrow{\alpha} D \simeq D\times\underline{1} \xrightarrow[(D\times Z)_i]{1_D\times Z_i=} D\times Z \xrightarrow{b^*} D\times Z \xrightarrow[(D\times Z)_f]{1_D\times Z_f=} D\times\underline{1} \simeq D \xrightarrow{\beta} Y$$

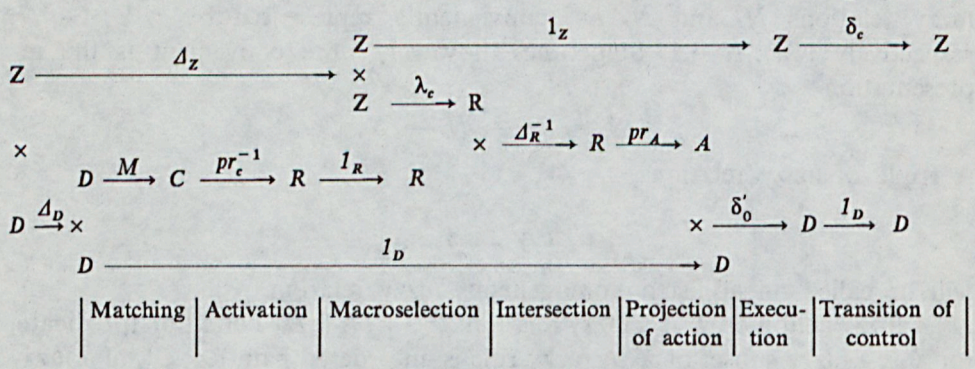
$$\lfloor\!\!\lfloor \quad B \quad \rfloor\!\!\rfloor$$

Fig. 2

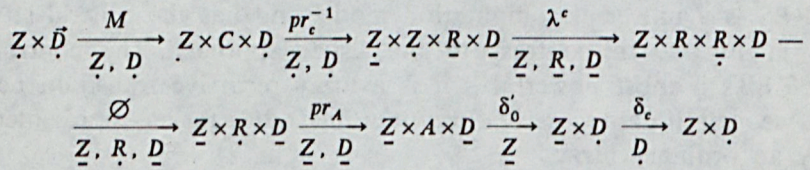
correspond to (1) and (2) and define two subsets of Z of initial and final states respectively. The essential part of the behaviour is B' and we shall consider only it hereafter. It is some sort (to be introduced later) of RTC of b.

Considering the one-step behaviour b, different equivalent representations can be deduced. Two of them are shown for example in Fig. 3a and Fig. 4a respectively. The first one corresponds to forward execution of rules with preconditions, with matching as first phase. The second one illustrates forward execution of rules with pre- or postconditions, with macroselection as first phase. Fig. 3b and Fig. 4b show the same composition considering the pseudo-products as fuzzy relations with transit poles [10], [11].

Now we are ready to explore the RTC's in more detail and generality.

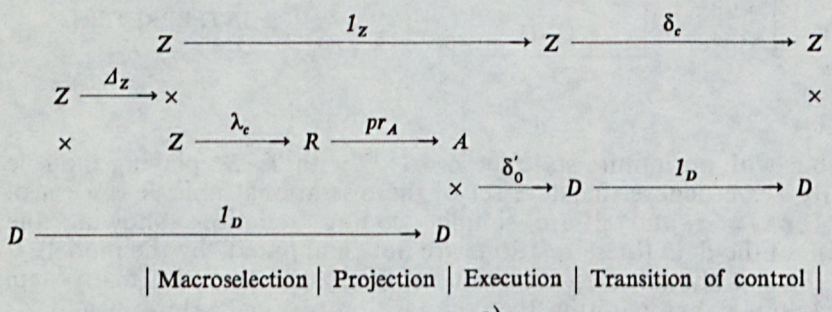


a)

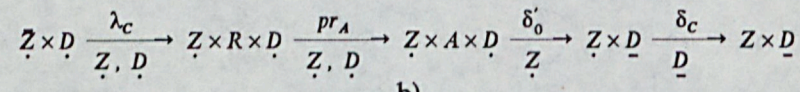


b)

Fig. 3



a)



b)

Fig. 4

3. The refllective transitive closure of a fuzzy relation or a fuzzy graph and its computation

Let $r: Y \to Y$ be an arbitrary homogeneous fuzzy relation. Its RTC, $r^*: Y \to Y$ is defined as

$$r^* = \bigoplus_{n=0}^{\infty} r^n,$$

where $r^0 = Id_y$, and for $n \geqslant 0$, $r^{n+1} = r \cdot r^n$. If $\{\varrho_i / i \in I\}$ is a family of fuzzy relations $Y \to Y$ then by $\bigoplus_{i \in I} \varrho_i$ we mean the fuzzy relation $Y \to Y$ defined by

$$\langle y, y' \rangle \mapsto \bigoplus_{i \in I} \varrho_i(y, y').$$

A discussion what an identity fuzzy relation Id_y is can be found in [13].

The RTC, r^*, of a homogeneous fuzzy relation, r, has been constructed in Fig. 5a.

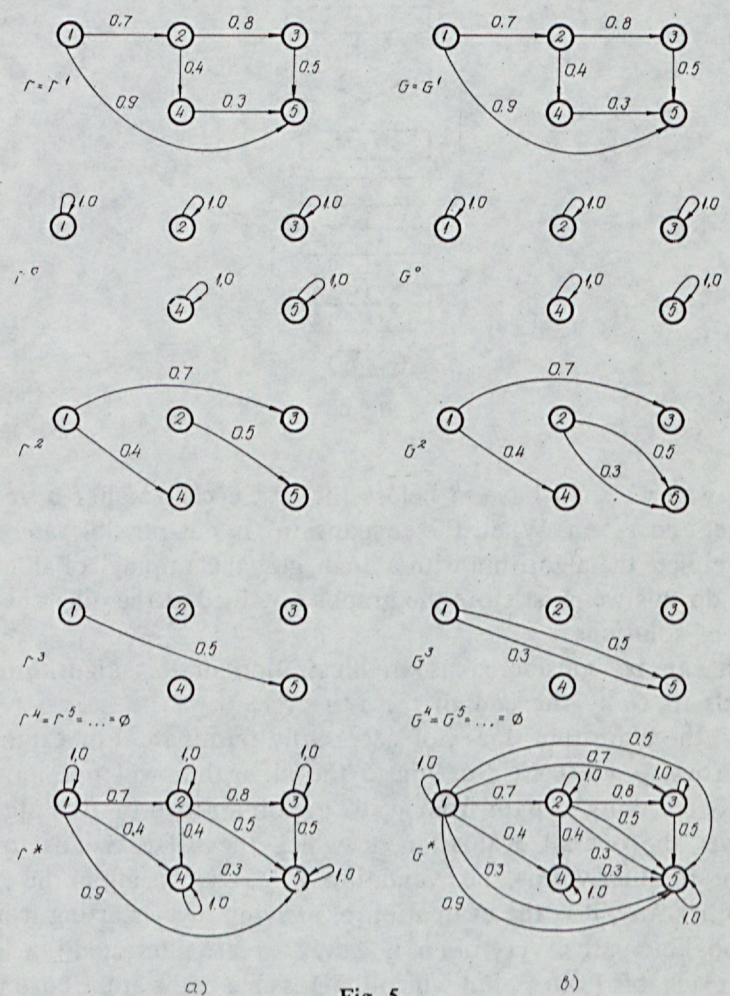


Fig. 5

A generalized algorithm for computing the RTC of a fuzzy relation $r: Y \to Y$ is shown in Fig. 6. Here G is a fuzzy set of paths in the graph of r. G is actually a fuzzy graph. The initial value of G contains exactly the paths of length 0, each path with valuation 1, i.e. this initial value is isomorphic to the set of nodes Y. At each step of the algorithm the following two actions are performed. Firstly, G is being „expanded" – each path of the former G is replaced by set of its successors, i.e. compositions of this path with an arrow of the graph r. The process of composing includes, of course, the calculation of the valuation degree of the composition arrow using the formula $v_c = v_1 \square v_2$ *where* v_1 *and* v_2 are the degrees of the arrows being composed. Secondly, G is being „clamped" i.e. each class of parallel arrows in G is replaced by one arrow – their „sum". The sum arrow has a valuation degree obtained through the operation $\oplus$ on the degrees of all the arrows in the class.

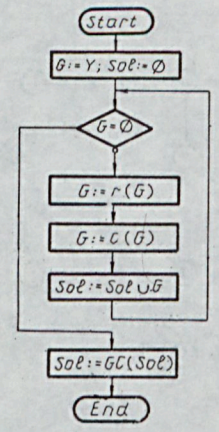


Fig. 6

In this way we shall have r^0 before the first cycle, r^1 after it, r^2 after the second one, ano so on. When the „expansion" is not possible any more, we have to complete the algorithm with a final „global clamping" of all graphs. To be able to do this we must store the graphs obtained at the different cycles, in a set Sol of solutions.

Fig. 5a can be considered as an illustration of this algorithm. Here r^i corresponds to G by the end of the i-th cycle.

Clearly, the algorithm does not necessarily terminate. For example, if we add an arrow ②⟶① to r of Fig. 5 the algorithm will not terminate.

Roughly speaking, a path from y_1 to y_2, obtained using the algorithm of Fig. 6, gives the overall evaluation, v, which the RTC assigns to the pair $\langle y_1, y_2 \rangle$ or, in other words, the evaluation of $\langle y_1, y_2 \rangle$ being in the RTC. Said in yet another way, it is the evaluation of arriving in y_2 starting from y_1 and using all possible paths. Very often, however, we are interested not just which nodes are reachable from y_1 but which the possible paths are. Hence we do not

want to clamp the nodes as in the algorithm of Fig. 6. The algorithm takes the form shown in Fig. 7. An element in the final value of G corresponds now to one particular path giving also its valuation. For the example of Fig. 5a r^* will take form shown in Fig. 5b.

The graph obtained through this modified algorithm will be called the RTC of the graph corresponding to the initial fuzzy relation. We have to notice that it is not simple. If we perform global clamping to a RTC of a simple graph we obtain the RTC of the fuzzy relation corresponding to this graph.

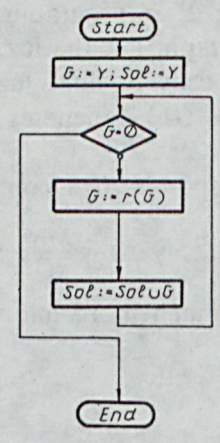


Fig. 7

Obviously the algorithm of Fig. 7 fits also the case when instead of r we have a graph which is not necessarily simple. So the flowchart of Fig. 7 is in fact an algorithm for „computing" the RTC of a fuzzy graph. To be more precise, let us introduce the formal definition of this notion. If $r_1=\langle N, A_1, s_1, t_1\rangle$ and $r_2=\langle N, A_2, s_2, t_2\rangle$ are two fuzzy graphs with coinciding sets of nodes then their composition, $r_2 \circ r_1$, is a fuzzy graph $\langle N, A, s, t\rangle$ where to every pair of arrows $a_1 \in A_1$ and $a_2 \in A_2$,

$$n_1 \xrightarrow{a_1 : v_1} n_2 \xrightarrow{a_2 : a_2} n_2$$

such that $t_1(a_1)=s_2(a_2)$ an arrow $a_{1,2}$ is assigned in A,

$$n_1 \xrightarrow{a_{1,2} : v_{1,2}} n_2$$

with $v_1{}'_2=v_1 \square v_2$. Now we define $r^0=Id_N$, $r^{n+1}=r \circ r^n$ and

$$r^* = \coprod_{n=0}^{\infty} r.$$

The „nonintersecting union" G of two fuzzy graphs G_1 and G_2 is another one with the same node set and $A=A_1 \amalg A_2$, i.e. there is an arrow in A for every arrow in A_1 or A_2, keeping the valuation degree unchanged.

The RTC's of fuzzy graphs are more important for us than the RTC's of homogeneous fuzzy relations because of our orientation to fuzzy RBS interpreters. That is why we continue our consideration for the former case only.

4. RTC of an adjusted fuzzy graph

More interesting for our purposes is a notion which we shall name RTC of an adjusted fuzzy graph (cf. Fig. 2). In programming this problem is known as searching in, or traversing a graph. Let the fuzzy graph has node set Y and X and Z are its initial, respectively final fuzzy subsets. The generalized algorithm is shown in Fig. 8. $G \cap Z$ denotes the composition of G with Z considered as the graph of $Z^{(3)}$.

The solution set Sol corresponds to a composition of the form

$$Y \xrightarrow{X^{(3)}} Y \xrightarrow{r} Y \xrightarrow{Z^{(3)}} Y.$$

The former, however, carries much more information.

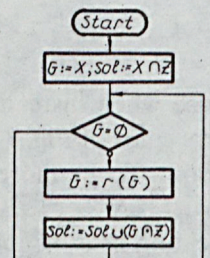


Fig. 8

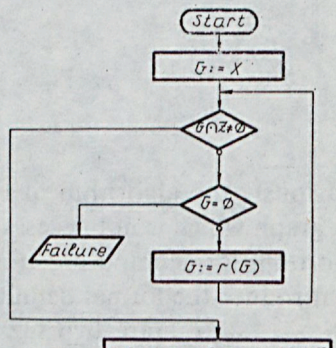


Fig. 9

The RTC of a fuzzy graph (Fig. 6) is a special case of the RTC of an adjusted fuzzy graph (Fig. 8) if we choose $X=Z=Id_Y$.
If we are looking for only one solution path then the algorithm of Fig. 9 should be used.

5. Backtracking

The algorithms of Fig. 6-9 cannot be executed on a conventional sequential computer. These intrinsically parallel computations must be simulated. As is well-known, there are two approaches to this simulation – constructing

a solution graph (search graph), and backtracking. Hereafter we shall discuss backtracking strategies only. The main advantage of this approach is the weaker memory requirements. Moreover, in contrast to the usually developed algorithms, ours are nor recursive. We consider here blind search only.

Now we shall move away from the generalized parallel algorithms and consider a sequential analogue to the algorithm of Fig. 8. It is depicted in Fig. 10. For the sake of simplicity X and Z are supposed to be crisp sets.

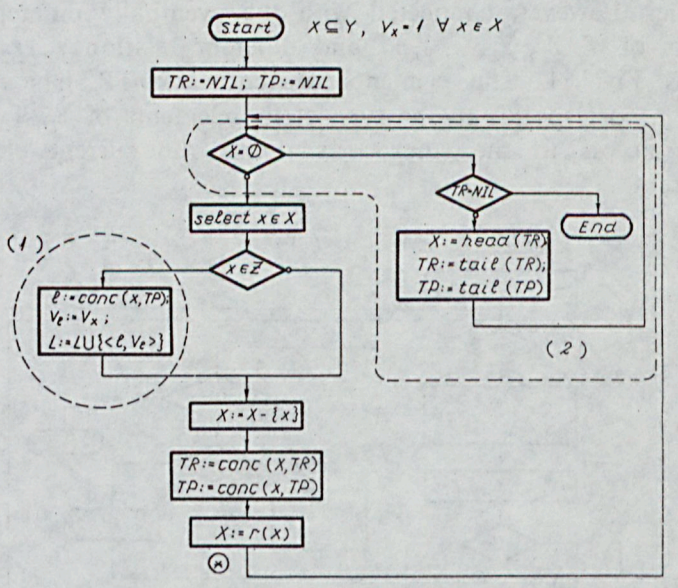


Fig. 10

TR is a stack of „reserves". Formally it is considered as a list. TP is a stack of nodes. At a moment when a current node x occurs to be final then the concatenation of the contents of TP and x is a successful path which has to be fixed as a solution in L. When executing $X := r(x)$ the valuation of every element of X is to be computed as well, using the valuation of x. It may be convenient if X is ordered with respect to the valuations of its elements, with this order being used during the selection $x \in X$. X is stored into the stack TR as a fuzzy set, too.

We found it more convenient instead of determining Z and realizing the test $x \in Z$ to modify the graph such that the final nodes are replaced by usual nodes and special „ final" arrows:

$$n \xrightarrow{\cdot}$$

The implementation of $r(x)$ is also modified: the effect of part (1) (the test for success) of Fig. 10 is now performed by r when searching for the successors of n. Hereafter we shall omit tests for success in our algorithms.

If the test $X = \emptyset$ for the initial value of X is not needed in the context of a particular application then part (2) can be moved to point ⊛ in the algorithm.

6. The case of composite fuzzy relation

Very often in our applications the fuzzy relation $r: Y \to Y$ is itself a composition of component fuzzy relations:

$$Y \xrightarrow{r_1} Y_1 \xrightarrow{r_2} Y_2 \to \cdots \to Y_{n-1} \xrightarrow{r_n} Y. \tag{4}$$

The idea behind the backtracking strategy remains the same. There are difficulties, however, connected with the eventually different types of the elements of $Y, Y_1, Y_2, \ldots, Y_{n-1}$ and different relation $r_1, r_2, \ldots, r_n$. An algorithm is Fig. 11. The elements of the stack TR have n different fields. Field f_i corresponds to the type of the elements of Y_i. Two pointers control the access to the stack – p points the current element, and i – the current

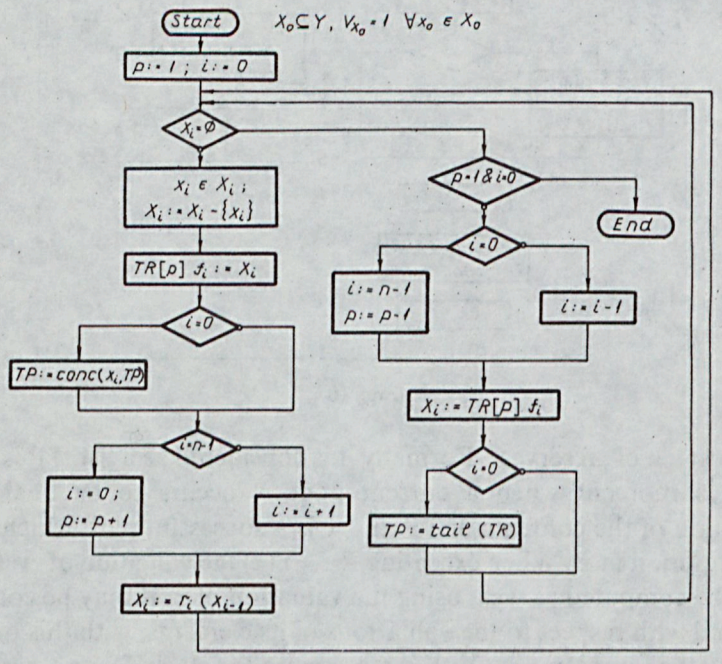


Fig. 11

field in the element. It is supposed that when expressing a successful path we are interested in the elements of Y only – so the stack TP includes only elements of level 0. By TR $[p] \cdot f_i$ the field f_i of the element with pointer p of the stack TR is denoted.

The algorithm of Fig. 11 can be used as a basis for the implementation of a universal program for obtaining the RTC of a fuzzy graph for any composition (4). For each particular case one has to define the number of levels n, the type of x_i (X_i, TR and TP will be automatically known) and the procedure r_i *for* each i, $0 \leqslant i \leqslant n-1$.

For any particular case, however, it would be probably more effective to develop a particular algorithm using the ideology underlying Fig. 11. The „particularity" concerns the number of levels and the kind of the relations. Fig. 12 exposes an algorithm for the case $n=3$.

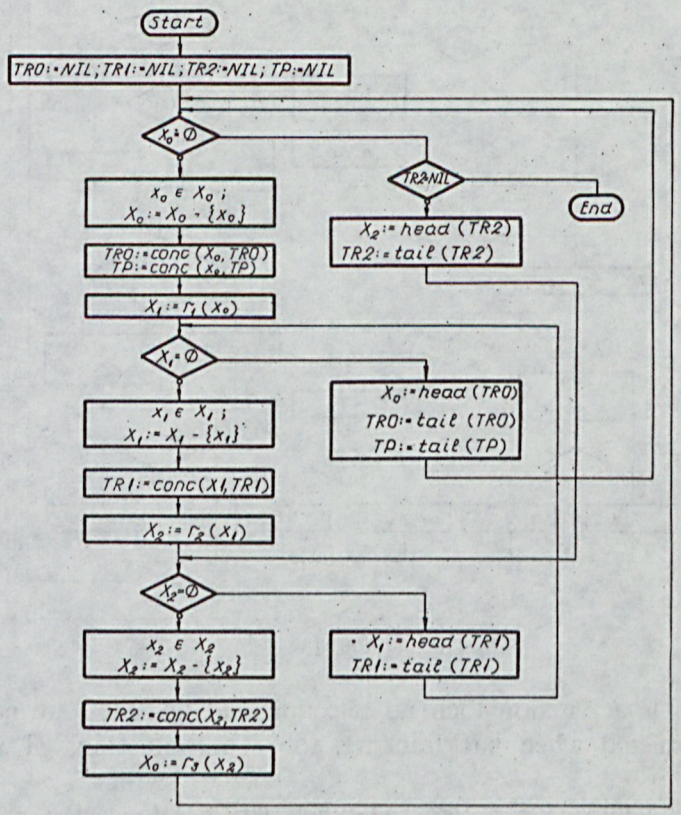


Fig. 12

Avery important particular kind of fuzzy relations is the case when r_i for some indexes ire partial functions. The algorithms of Fig. 13 and Fig. 14 are related to this case.

In Fig. 12 the failure at a level returns the process to the previous level. Stack TP corresponds to level 0. An exit is possible from level 0 only.

Consider now the case when r_i for each $i \in I'$ is a partial map. Taking this into consideration we can simplify the algorithm making it more effective (faster and requiring less memory). A modified universal algorithm is shown in Fig. 13. (cf. Fig. 11). Here e is a flag. Its value is true if $X_i=\emptyset$ or $r_i(x_{i-1})$ is undefined (not completed successfully). It is admitted that e is formed automatically by each r_i.

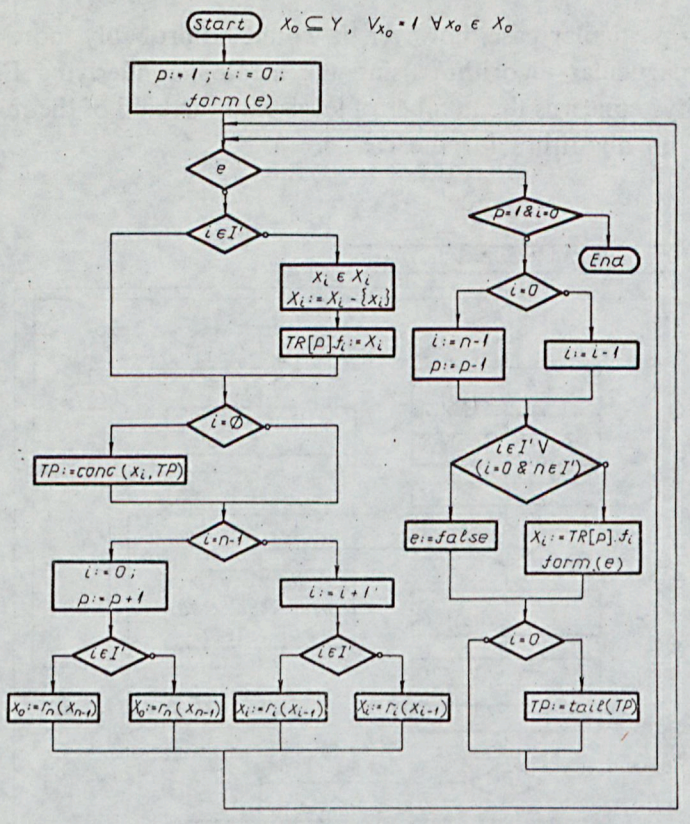


Fig. 13

When r_i is a function then no selection and no stack are needed. Such a level is missed when backtracking, too. The valuations of x_{i-1} and x_i coincide.

These conclusions can be used when writing down the argorithm for a concrete case as well. An analogue to the algorithm of Fig. 12 is shown in Fig. 14 supposing r_3 is a partial function. The only step aside from the general algorithm is that here we prefer the test of e to be at the very end instead from being at the very beginning because we suppose e is always true for the initial x_0.

7. The case of fuzzy relations with transit poles

Examining Fig. 3 and Fig. 4 we see that as a rule the one-step behaviour of a RBS consists of component fuzzy relations with transit poles. Consider, for example, the phase macroselection of Fig. 4. The corresponding fuzzy relation is $1_z \times \lambda_C \times 1_D$. The usage of the concept of a fuzzy relation with transit poles gives us the opportunity of considering λ_C only instead of the whole pseudoproduct. We have only to slightly modify the algorithms storing in the

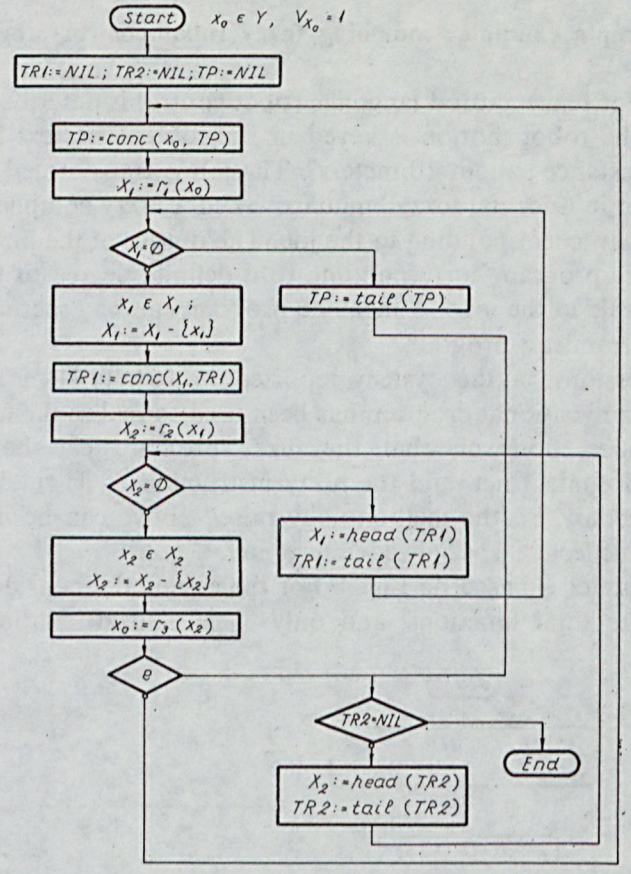


Fig. 14

stack not only a subset of R of reserves but the elements corresponding to the transit poles as well. For our example we have to store $z \in Z$ selected before λ_C, and the current state $d \in D$ of the data (the transit poles are Z and D).

In principle, instead of using a stack for storing some X_i, it is possible to avoid this by performing inverse actions when backtracking (the usage of stack requires memory and time). This approach has been widely used in our universal crisp interpreters (e.g. [4]-[6]). Applying this method for a fuzzy relation with transit poles we deduce that the values corresponding to the transit poles do not have to be restored from a stack or by inverse execution since the action on them is an identity. In fact, if we have the situation

$$A \times Y_{i-1} \xrightarrow{r_i} A \times Y_i \xrightarrow{r_{i+1}} A \times Y_{i+1} \xrightarrow{r_{i+2}} Y_{i+2},$$

If r_i and r_{i+1} are fuzzy relations then we do not need to store A on the levels $i-1$ and i at all. If, however, r_{i+1} is a function there will be no stack on level $i+1$ and we backtrack from level $i+2$ to level i directly. The element of A has been meanwhile changed by r_{i+2}. Hence in this case A has to be included into the stack of level i (it is not needed for level $i-1$).

8. Simple example: compiling fuzzy robot control programs

A system for fuzzy natural language robot control is described in [6], [14]. The job for the robot motion is keyed in Bulgarian. The text includes fuzzy notions, for instance „about 10 meters". The job is transformed into a formal fuzzy program in a formal fuzzy language. Then a fuzzy compiler tries to find a path on a map corresponding to the job. The output of the fuzzy compiler is a deterministic program corresponding to a definite degree to the fuzzy one, and conformable to the map. A machine program can be generated implementing the deterministic program.

In the versions of the system reported in [6], [14], a comparatively complicated conventional program has been used to realize the fuzzy compiler. One can notice, however, that the fuzzy program can be regarded as a degenerated control net, and the fuzzy instructions – as rules consisting of single actions only. So the methodology raised above can be used to obtain a simple and effective specialized interpreter.

The interpreter is based on Fig. 4. For the case under consideration all λ_C, pr_A and δ_C are total functions, and only δ'_0 is a fuzzy relation.

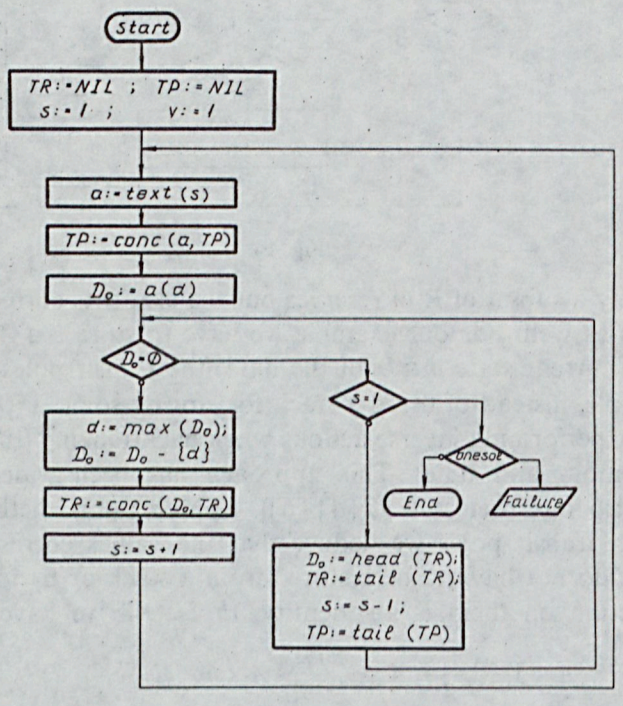


Fig. 15

The corresponding algorithm is shown in Fig. 15. On the phase of selection an element of D_0 is chosen with a maximal valuation degree (an element of D_0 is a complex of coordinates, direction and current valuation). Value true of flag

„onesol" means we are looking for one solution only. We take into consideration the fact that fuzzy program is linear, so instead of pulling the number, s, of the instruction from the stack TR during backtracking (cf. section 7) we reconstruct it by executing $s := s-1$.

References

[1] K. Kratchanov, Computations, Problem Solving and Rule-Based Systems, CSTR 33/85, Brunel Univ., 1985, 203 pp.

[2] Id. On the Foundations of Rule-Based Systems, CSTR 34/85, Brunel Univ., 1985, 47 pp.

[3] K. Kratchanov at al., Interpreter with Backtracking of Controlled Rule-Based System, Techn. Univ. Russe Annual 27, ser. 3, 1985, 33-37 (in Bulgarian).

[4] K. Kratchanov et al., An Intelligent System for Solving Problems of Movement Difined in Bulgarian. Ibid., 28, ser. 6, 1986, 55-60 (in Bulgarian).

[5] K. Kratchanov et al., Mathematical Model of a RBS and its Implementations. East-European Category Seminar, Predela 1987, Summaries, 19-20.

[6] K. Kratchanov, I. Stanev, A RBS for Natural Language Robot Control. Proc. 4-th Intern. Conf. of Artificial Intelligence and Information-Control Systems of Robots, Smolenice, Czechoslovakia, 1987 (I. Plander – ed.), North Holland 1987, 304-309..

[7] K. Kratchanov, Towards the Fundamentals of Fuzzy RBS's, CSTR 35/85, Brunel Univ. 1985, 19 pp.

[8] K. Kratchanov, I. Stanev, Fuzzy RBS's, Fuzzy Algorithms, Fuzzy Reasoning. First Joint IFSA–EC & EURO–WG Workshop on Progress in Fuzzy Sets in Europe, Warsaw 1986, (J. Kacprzyk and A. Straszak – eds.), Prace IBS PAN, 167, Warszawa 1988, 155-166.

[9] K. Kratchanov, On Rules in RBS's. East-European Category Seminar, Predela 1986, Summaries, 9-10.

[10] K. Kratchanov, Relations and Fuzzy Relations with Transit Poles, CSTR 32/85, Brunel Univ., 1985, 46 pp.

[11] Id., Fuzzy Relations with Transit Poles and Their Category-Theoretical Consequences, East-European Category Seminar, Predela 1987, Summaries, 21-22.

[12] K. Kratchanov et al. On the Backtracking Strategy, Techn. Univ. Russe Annual 27 (1985) ser. 3, 111-116 (in Bulgarian).

[13] K. Kratchanov, On Fuzzy Relation Formalism and Corresponding Category-Like Structures. East-European Category Seminar, 1986, Summaries, 11-12.

[14] I. Stanev et al., Inteligent System for Fuzzy Robot Control. Techn. Univ. Russe Annual 28, 1986, ser. 6, 43-48 (in Bulgarian).