A Program System For Generalized Nets Models Constructing

Dilyana Budakova, Lyudmil Dakovski

Abstract. *In this paper is presented a program system for building Generalized Nets (GNs) models of average complexity. The possibilities and some special features of the system are discussed. The GUI-based system provides a graphical visualization of the process of GNs constructing. The program composes and displays on the screen the respective indexed matrix of the GN's transitions, and allows the user to choose the predicates for a token's transfer from an input to a respective output position.*

Keywords*: generalized nets, modeling, program system.*

INTRODUCTION

The Generalized Nets [1] are a contemporary and widely approved mathematical formalism for process modeling in different areas. Some of its advantages are the possibility for representing the progress of states and processes achieving a compactness and unification of the models, the ability for parallel processing of the predicates of each position. As much are the transitions and positions defined as simple will be the predicates which control the tokens' transfer. And vice versa - the less transitions and positions are defined the more complicated the predicates will be.

In [2, 3] is discussed another program realization of GNs, and in [4] are described some program aspects of the theory of Generalized Nets.

In this paper we present the developed program system for GNs models constructing. The system enables a graphical visualization of the GNs building process conducted by a dialogue with the user. The respective indexed transition matrixes are automatically composed and visualized. The user is asked to choose the respective predicates, which defines the rules for token's transfer from a given input position to another output position. The program system described here is implemented for series of experiments in the intelligent agent's behavioral aspects research [5,7] and in a research for automated composing of verses [6]. An example of a graphically build GN is depicted in fig.1.

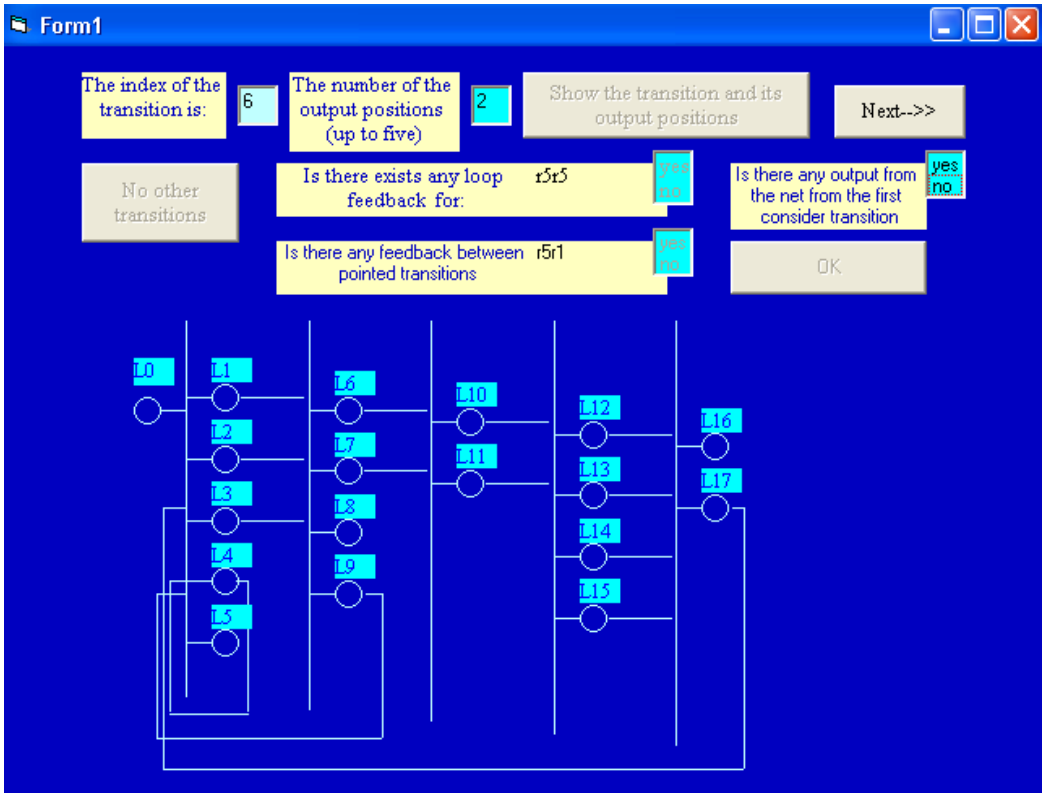

Fig.1 A graphical representation of a GNs model.

HOW TO USE THE PROGRAM SYSTEM. MAIN FORMS. GNS MODEL BUILDING PHASES.

Form 1. Visualization of a GN.

In fig.2 is depicted the first program form, appearing immediately after its start. A transition with one input and one output position is given. The user has to specify (and to enter in the respective text box) the desired number of the output positions (three in our example) for the considered transition. After clicking on the button "Show the transition and its output positions" they are drown by the program.

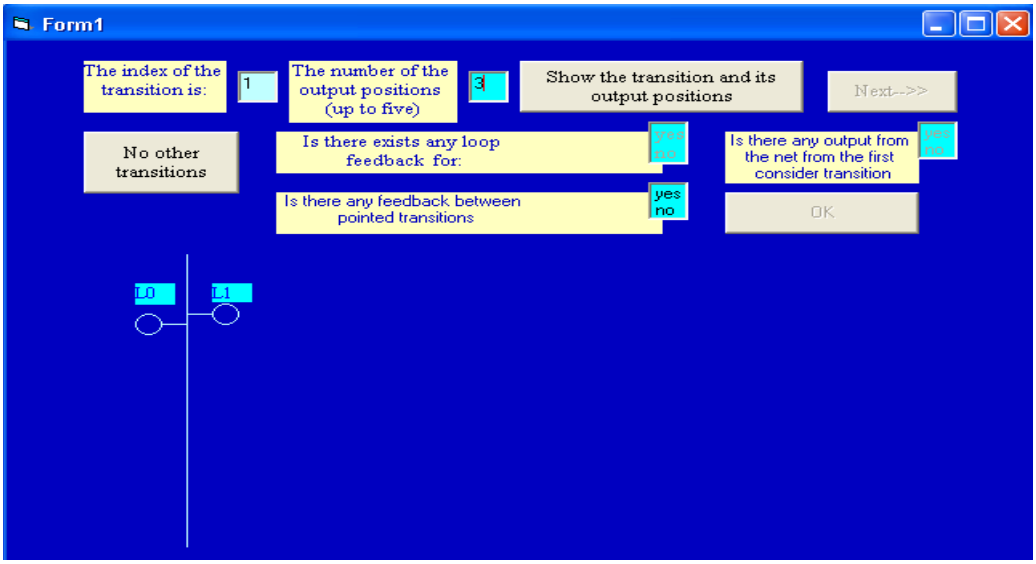

Fig. 2 The first form of the program system.

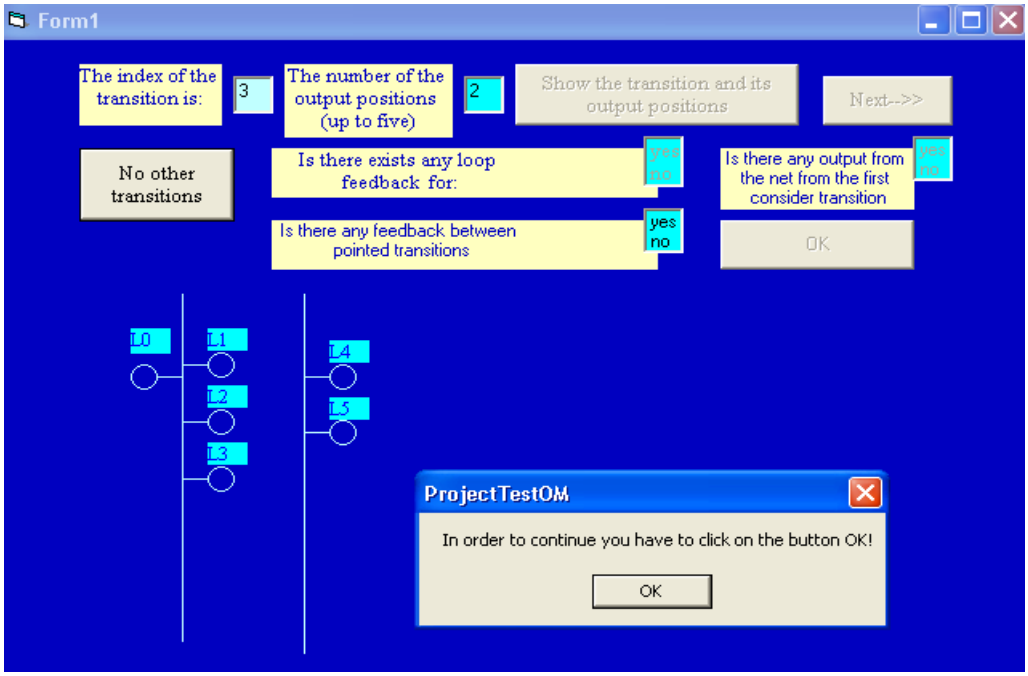

Fig. 3 Finalizing the constructing of the transitions and their output positions by clicking on the button "No other transitions"

After that the user is asked to enter the number of the output positions (two, in our example) for the second transition. The button "Show the transition and its output positions" is clicked again and the program illustrates the transition and its output positions.

Let assume that (for the example given) no more transitions are needed. After a click on the button labeled "No other transitions" a message box is generated. It informs the

user that in order to continue, she/he have to click on the button "OK". This is required in order to disable the buttons, which were used until that moment, and to enable the rest of the buttons (which were disabled by that moment).

The next step in the building process is to answer to the program questions concerning the feedbacks to every transition. Accounting the user's response the system will visualize the respective feedbacks. First must be specified if there will exists any feedback to and from the transition concerned. For instance, in the example given, there could be a loop feedback from the first transition r1 to the first transition r1 (i.e. to itself). If the user decides, that such a feedback won't exist she/he has to mark "No" in the respective listbox. It must be specified that there will be no output position for this transition .

As the transition discussed above (r1) is the initial (there are not any preceding transitions) and there exists no feedback from any of its outputs, and also there are not any net's outputs, consequently the three outputs are connected directly to the second transition. These connections are generated and depicted automatically by the program (accounting the user's answers). The next question is if there exists any loop feedback (r2r2) and if there are any outputs from the net. For the example given, the answer is No (no feedback and no outputs).

As the output positions of the transition considered are two, and as a preceding transition exists, then, the program asks the user to specify if there is any feedback from the second transition r2 to the first transition r1. Let assume that for the example considered the answer will be yes. The button "Ok" is clicked, and a message is appeared informing that as the last GN's transition is considering and as there are no more preceding transitions for specifying, thus the free position will be considered as an output one. The feedback is generated automatically by the program and the "Next→" button becomes available (see Fig.3).

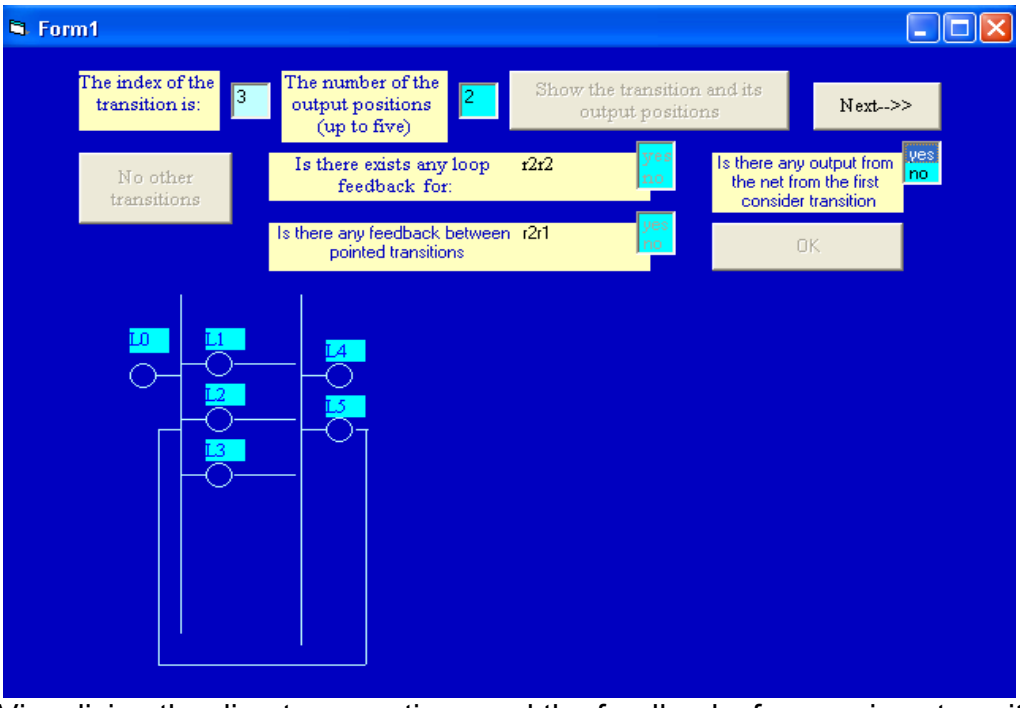

Fig. 3 Visualizing the direct connections and the feedbacks from a given transition (r2 in our example) to the preceding transitions (r1 in this case).

Form 2. Computing the indexed matrix of the transitions in the constructed GN.

The "Next→" button is clicked and in result of the execution of its event 'click' the second form of the program is visualized (fig. 4). That form generates the respective indexed matrix of the GN build so far.

The user has to click on the button "Show the transitions matrix" and they will be generated and visualized automatically by the program. By default, some predicate corresponds to each pair of input – output positions. A message is shown informing the user that she/he could use the keys: backspace - for deleting; arrows – for moving the cursor up, down, left or right; t – to assign a true value to the respective predicate; f – to assign a false value to the respective predicate; p - to mark that in that place must be a predicate, if the user is written down false or true by mistake. To edit the values a single click on the respective button and positioning (by the arrows keys) to the exact cell of the table is enough.

For the example given nothing will be changed in the indexed transitions matrixes.

The program could visualize GNs with maximum nine transitions and five output positions for a transition (because this is the maximum number which could be visualized on a single screen). Respectively, a nine indexed transition matrix (together with their input and output positions) could be visualized in a single screen. It's taken into account that if a transitions feedbacks exists then the number of the input positions, and hence the table size will grow. The number of the rows in each table corresponds to the number of the input positions of the respective transition.

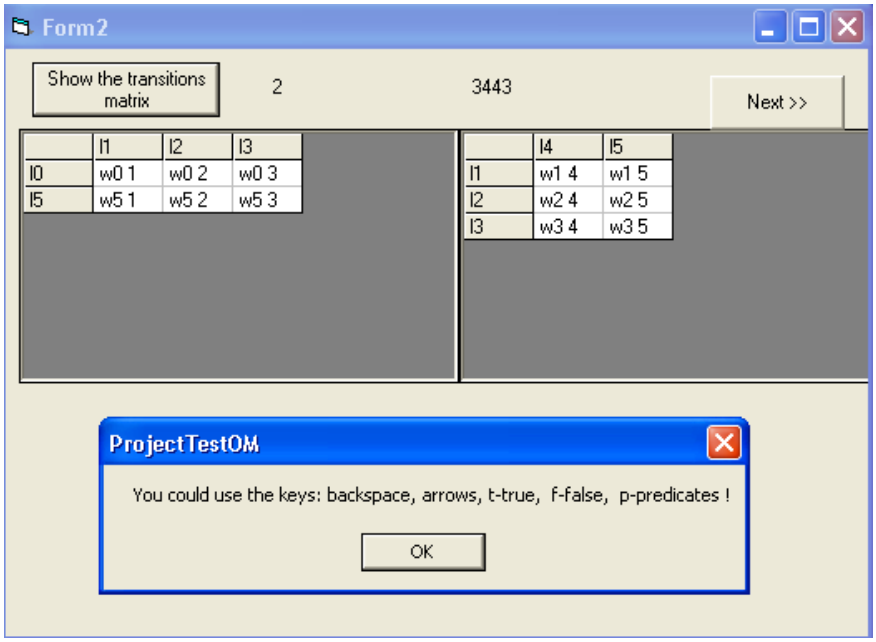

Fig. 4 A visualization of the respective indexed matrixes of the existent transitions.

Form 3. Choosing the subroutines, which implements the predicates for a token's transfer from a given input to the respective output position.

The third main form of the program, labeled "Predicates", is visualized after clicking on the button "Next→" (fig.4). In the beginning it is empty – the user has to click consequently on the buttons "Show the predicates table" and "Show the reference table". The predicates table consists of as many rows as many are the predicates of this Generalized Net. Each row represents only one predicate and is labeled with the predicate name (for instance w01, w02, w03 …. – these are the predicates for a token's transfer from an input position 0 to the output position 1, 2, and 3 respectively). The first column of the table contains the transition's index, the second one – the index of the input position, and the third one – the index of the output position. In the forth column must be entered the name of the subroutine which implements the respective predicate. Ready-to-use subroutines, implementing various predicates are developed. The desired one has to be selected from the first listbox (placed beneath the button "Show the predicates table"). After selecting the proper subroutine the user has to click on that cell (in the predicates

table), which is in the column labeled "Methods" and in the desired row. The name of the selected method will appear automatically in the cell.

The next columns of this table are: "If True", "If False". By specifying their values the user can set some of the input arguments of the subroutine (the method which implements the respective predicate) to take on true or false values. The arguments are selected from the second listbox.

If the user enters some wrong value by mistake, it could be easily corrected just by selecting again a predicate name from the list and clicking on the desired position. The value will be refreshed automatically. An argument placed in a cell of column "If True" is interpreted as a true value. Analogically, the arguments in the column "If False" are interpreted as a false. The program would not allow a predicate name to be entered in the arguments columns and vice versa.

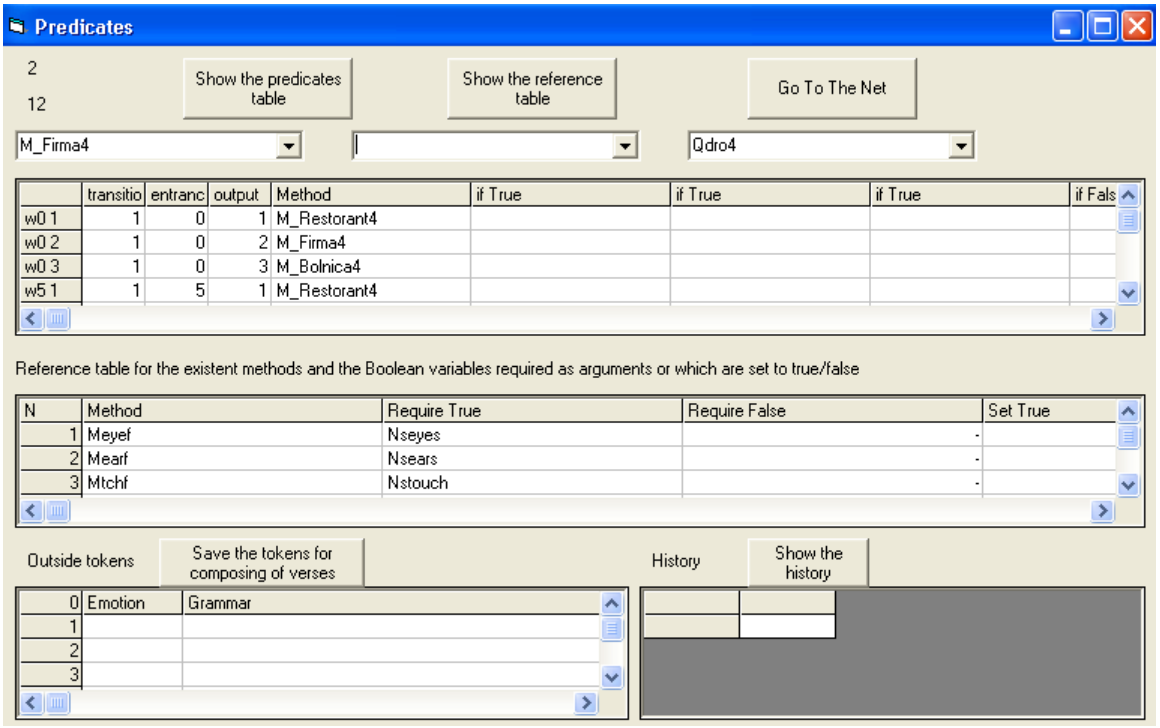

Fig. 5 Specifying the subroutines, which will implements the predicates needed for the GNs model realization.

But how the user will know which the developed methods are, what are their arguments, and what could be the requirements for these arguments? In order to answer these questions a second table, labeled "Reference table for the existent methods and the Boolean variables required as arguments or which are set to true/false"

The third listbox contains the list of tokens. For each experiment a proper token must be selected. A token could be considered as a whole GN, but in the simplest case a token is implemented as a relational table containing the required characteristics. For instance, each row of such a table could represent a single token.

For the example considered before are used predicates with the names: M_Restorant4, M_Firma4, M_Bolnica4, false, true.

After filling in the predicates names and after selecting Qdro4 the experiment could be started by clicking on the button "Go To The Net". The program begins to process the data. In the runtime, a proper messages are displayed which informs the user, for instance that the program has reached to certain time-interval. The last message is that the program is ready and the experiment is finished successfully "I am ready!"

CONCLUSION

A program system is developed which enables its users to design a Generalized Nets models. The program allows a graphical visualization of the constructed GNs – the transitions, the positions, and the connections between them. Because of the screen limitations, aiming to display the whole graphic and indexed transitions matrixes in a single screen, it's chosen to work with array with maximum nine transitions and five positions for each transition. If the ability to scroll the image on the screen is added, then only by re-declaring the array the program will be able to visualize a GNs with arbitrary numbers of transition and arbitrary numbers input and output positions. The transitions and the connections between them are constructed according to the dialog with the user. The program generates automatically and visualizes the indexed transitions matrixes and the table containing the indexes of all of the predicates in the constructed GN. The user can select (from the predicates listbox) the desired subroutines for implementing the predicates which controls the token's transfer from an input to an output position, and the tokens type as well. The requirements for these ready-to-use subroutines could be read in the table labeled "Reference table for the existent methods and the Boolean variables required as arguments or which are set to true/false". The system enables the user to create its own subroutines, as well as its own tokens type. Another program in the system gives information about a token position in given moment of time, and where the token could be positioned. Each time-interval is divided into two subintervals: the first is for token positioning, the second – for information exchange and decision making.

The idea underlined in the presented program system is to implement the theory of the Generalized Nets. The developed system is suitable for research in the dynamic processes and their states in time

It gives the advantage of a compact models representation, achieving its unification, and the ability for parallel processing of the predicates for each position.

REFERENCES

[1] Atanassov K. , Generalized nets, World Scientific Publ. Co., Singapore, 1991

[2] Atanassov, Krassimir T.; Christov, Rumen St. Program Package for Generalized Nets (PPGN). In: *Petri Net Newsletter No. 42*, pages 23-26. August 1992.

[3] Atanassov, K.T., Chistov R.S., About the program realization of the generalized nets. In: *Advances in Modeling & Analysis, AMSE Press, Vol. 17, No. 1* 1993.

[4] Atanassov, K.T.; Yanev, K.D.; Atanassova, L.C. Theory of Generalized Nets (A Programming Aspect). In: *Proc. 2nd Symp. on Automation of Scientific Research, Varna, May 1983*, pages 397-399. 1983.

[5] Budakova D., Dakovski L. Modeling a Cognitive-Emotional Analysis and Behavior, Information Technologies and Control, pages 33-29, YEAR III, 3/2005.

[6] Budakova Dilyana, Dakovski Lyudmil. Generalized net model of a program system for composing verses. Proceedings of the International Conference on Computer Systems and Technologies (e-Learning) CompSysTech'04, Rousse, Bulgaria, 17-18 June, pages II.10-1 – II.10-6, 2004.

[7] Budakova D., Dakovski L. Generalized Net Model of Behavior. International Conference Automatics and Informatics'04.Sofia, Bulgaria, October 6-8, 2004, p21–24.

ABOUT THE AUTHORS

Eng. Dilyana Budakova, M.Sc, Assistant at the Department of Computer Systems, Technical University of Plovdiv. St. Peterburg Blvd. 61, 4000 Plovdiv, Bulgaria. E-mail: dilyana_budakova@yahoo.com

Eng. Lyudmil Dakovski, Prof. D.Sc. CLBME – BAN. Centre of Biomedical Engineering Bulgarian Academy of Sciences. Acad. G. Bonchev Str., bl. 105, 1113 Sofia, Bulgaria. E-mail: lgd@clbme.bas.bg