BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **7**, No 1

Sofia • 2007

Modification of the Algorithm for Token Transfer in Generalized Nets

Krassimir Atanassov[1], *Violeta Tasseva*[1], *Trifon Trifonov*[2]

[1] *Centre for Biomedical Engineering, 1113 Sofia*
E-mails: krat@bas.bg violeta@clbme.bas.bg
[2] *Faculty of Mathematics and Informatics, Sofia University, 5 James Bourchier Str.,*
E-mail: triffon@fmi.uni-sofia.bg

Abstract: Generalized Nets (GNs) *([1]) are extensions of the Petri nets [2] and other modifications of theirs. They are tools intended for detailed modelling of parallel processes.*

The movement of the tokens in each GN is executed by preliminary defined algorithm ([1, 3]). The completely defined steps of the model execution guarantee its correctness, prevention from critical situations and dead-locks. In the present paper is proposed a new, more effective algorithm for tokens' movement, which will increase the speed of the algorithm as a whole.

Keywords: *Generalized Nets (GNs), Petri nets, algorithm for tokens' movement.*

1. On the concept of Generalized Nets

A generalized net is a collection of ***transitions***, defined in turn as a set of ***places*** (see Fig.1). For each transition there is an index matrix with elements – predicates. Some GN-places contain ***tokens*** – dynamic elements entering the net with initial characteristics and getting next ones during their movement in the net. Tokens proceed from the input to the output places of the transitions if the predicate corresponding to these places is evaluated as *"true"*. Every token has its own identifier and collects its own history that may influence the development of the whole process modeled by the generalized nets.

Two time-moments are specified for the generalized nets: startup and termination of functioning, respectively.

A generalized net could have only a part of its components. In this case it is called a reduced GN. Here we shall give formal definition of a reduced generalized net without temporal components, place and arc capacities, and token, place and transition priorities.

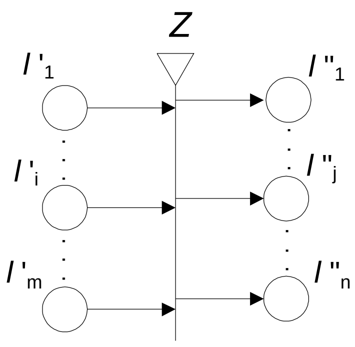

Fig. 1. GN-transition

Formally, every transition in the used below reduced generalized net is described by a three-tuple:

$$Z= < L', L'', r >,$$

where

a) L' and L'' are finite, non-empty sets of places (the transition's input and output places, respectively); for the transition these are

$$L' = \{ l'_1, l'_2, ..., l'_m \} \text{ and } L'' = \{ l''_1, l''_2, ..., l''_n \};$$

b) r is the transition's *condition* determining which tokens will pass (or *transfer* from the transition's inputs to its outputs; it has the form of an Index Matrix (IM):

$$r = \begin{array}{c|c} & l''_1 \ \ldots \ l''_j \ \ldots \ l''_n \\ \hline l'_1 & \\ \ldots & r_{i,j} \\ l'_i & (r_{i,j} \ - \ \text{predicate}) \\ \ldots & (1 \le i \le m, \ 1 \le j \le n) \\ l'_m & \end{array};$$

r_{ij} is the predicate that corresponds to the i-th input and j-th output place. When its truth-value is *"true"*, a token from the i-th input place transfers to the j-th output place; otherwise, this is not possible.

The ordered four-tuple

$$E = \langle A, K, X, \Phi \rangle$$

is called a *Generalized Net* if:

a) A is a set of transitions;

b) K is the set of the GN's tokens;

c) X is the set of all initial characteristics, which the tokens could obtain on entering the net;

d) Φ is the characteristic function that assigns new characteristics to every token when it makes the transfer from an input to an output place of a given transition.

A lot of operations (e.g., union, intersection and others), relations (e.g., inclusion, coincidence and others) and operators are defined over the generalized nets. Operators change the GN-forms, the strategies of token transfer and other. They are of six types: global, local, hierarchical, reducing, extending and dynamic operators.

2. The algorithm

In the present section of the paper the new algorithm for tokens' movement is represented.

2.1. Algorithm for transition functioning moment of time t_1

The simulation time-interval for the movement of the tokens with the highest priority from the input (one of a place) places to the corresponding output places is denoted with ***timestep***. The range of the timestep is t^0.

Step 1. The input and the output places are arranged by priority.

Step 2. In each input place are formed two lists: of all tokens, arranged by priority and an empty list.

Step 3. An empty index matrix R, which corresponds to the matrix of the predicates (r) of the given transition is generated. It is initiated with a value "0" (corresponding to value "false") of all of the elements of this new matrix, which:

– are placed in a row, corresponding to an empty input place,

– are placed in a column, corresponding to a full output place,

– are placed in (i, j)-th position, for which m_{ij}=0, i.e. the current capacity of the arc between i-th input and j-th output place is zero.

Step 4. The sorted places are passed sequentially by their priority starting with the place having the highest priority, which has at least one token and has not been passed on the current timestep. For its highest priority token (from the first list) the predicates corresponding to the relevant row of matrix R are checked. The elements of r, for which the elements of R are not zeros, are calculated.

Step 5. Depending on the execution of the operator for permission or prohibition of tokens splitting over the net, the token from Step 4 will pass either to all permitted to it output places or to that among them, which has the highest priority. If one token cannot not pass through a given transition on this tact, it is moved to the second list of tokens of the corresponding place. The tokens, which have entered into the place after the transition activation are moved into the second list, too.

Step 6. The capacities of all output for the transition places, which are input for other, active at the moment transitions, are increased with 1 for each token that has left them at this time step.

Step 7. The capacities of all output places, in which a token, determined in Step 4 has entered are decreased by 1. If the maximum number of tokens for a given output place is reached, the elements of the corresponding column of matrix R are made "0".

Step 8. The capacity of all arcs through which a token has passed is decreased with 1. If the capacity of an arc has reached 0, the elements of the corresponding output place of matrix R are made "0".

Step 9. The values of the characteristic function for the output places, in which tokens have entered (formed by the token passed according to Step 5) is calculated.

Step 10. If there are more places, which could be output for tokens at this step, the algorithm returns back to Step 4; on the opposite case it proceeds to Step11.

Step 11. The current model time t' is increased by t^0.

Step 12. Is the current time moment equal or greater to $t_1 + t_2$? If the answer of the question is "no" proceed to Step 4, else – termination of the transition functioning.

2.2. General algorithm for generalized net functioning in moment of time t_T

Step 1. Every token α, for which $q_k(\alpha) \leq T$ enters the net.

Step 2. It is constructed one abstract transition, whose sets of input and output places are empty at a moment of time t_T.

Step 3. It is determined all transitions, whose moment of activation t_1 is equal to the current time moment t_T.

Step 4. It is calculated the Boolean expression □ for each of the transitions, determined in Step 3.

Step 5. To the abstract transition are added all transitions from Step 3, whose Boolean expressions calculated Step 4 are "true".

Step 6. Over the abstract transition is executed the general algorithm for transition functioning for one time Step t^0.

Step 7. All transitions that have to be deactivated are removed from the abstract transition.

Step 8. The current model time is increased with t^0 ($t_T =: t_T + t^0$);

Step 9. It is checked if the current time moment is less then $T + t^*$; if the answer of the question is "yes" proceed to Step 3, else – it is gone to Step 10.

Step 10. End of the generalized net functioning.

3. Theorem

In the current section a theorem is formulated and proved, determining the advantages of the presented in Section 3 algorithm for GN functioning compared to the one, presented in [3].

Theorem. If a GN E is given, the results of its functioning compared to the presented in [3] algorithm and the one from Section 3, are equal.

P r o o f: Let a GN E be given. According to the Theorem for completeness of transitions [1] it is sufficient to show that the execution of each of the two algorithms over an unspecified transition from E gives one and the same result. Let us examine one transition Z of E. In accordance with both algorithms for transition functioning the input and the output places are arranged by priority; the tokens into the input places are ranged into a list by priority; Two empty index matrices R_1 and R_2 are generated, which correspond to matrix r of the predicates of the given transition (initially the elements of the matrices R_1 and R_2, for which are fulfilled the conditions from step 3, are made zeroes (corresponding to "false")). The arranged by priority places and tokens are passed equally by both algorithms with one difference – while the first algorithm continues calculating the non-zero elements of R_1, which correspond to predicates of r, the second one continues filling with "0" the elements of matrix R_2, which have no sense to be calculated in the future, because either the relevant output place is full or the arc capacity is zero, so the token could not pass. The tokens that can not pass by the second algorithm can not pass by the first one too, despite the fact that the corresponding predicate may be "true". Therefore, the obtained by the second algorithm results are the same as those of the first one, but they are achieved for equal or less time.

Besides, the calculation of tokens characteristics by the second algorithm is made right after the tokens movement. Knowing their priorities the order of their movements could be predicted. This can be applied for the calculation of the characteristic functions of the tokens, using preliminary calculated characteristics. For example, if we have three tokens, which ought to obtain the corresponding characteristics: A^B, $A^B.C$ and $A^B.C + D$, we could give the highest priority to the first token, lower priority to the second one and the lowest to the third token. In this case, if the first token obtains characteristic $X_1 = A^B$, the second one will obtain characteristic $X_2 = X_1.C$ and the third token – $X_3 = X_2 + D$. Therefore, bearing in mind that the time for exponentiation is greater than the time for multiplication, and the one for multiplication is significantly greater than the one for addition, thus the time for calculation of the three characteristics by the second algorithm is almost three times shorter than the needed for the first algorithm.

If the capacitates of the arcs and the places are infinitely large and there is no possibility to use preliminary calculated characteristics in the calculation of the values of next characteristics, both algorithms will function for the same time. But if the capacities of the places are limited and especially very small numbers, and preliminary calculated values could be used for the estimation of the next characteristics, the software implementation of the two transitions will be made in the worst situation for the same time, otherwise, obviously, the second algorithm will work faster. In the case of infinitely capacities of the arcs and the places and there are great number of tokens in the input places in every moment of time, the first algorithm will work faster, because it will not make the unnecessary in this case verifications, provided in the second algorithm.

4. Implementation

The algorithm presented here presents some optimizations to the original algorithm for token movement as given in [1]. For an object-oriented implementation of the algorithm in an interpreter for *GN*s instead of using the boolean matrix *R*, one would prefer local checking of the capacity conditions presented above at each place or arc. Since predicates in a predicate matrix are arbitrary in the general case, their evaluation should be considered a heavy operation. Therefore, predicates should be evaluated last, only when all other conditions for token movement are satisfied. The current implementation of an interpreter for *GN*s (GNTicker, [3]) already performs this check for a place capacity, and does not try to evaluate a predicate if the output place is full. The functionality regarding arc capacities is yet to be implemented.

R e f e r e n c e s

1. A t a n a s s o v, K. Generalized Nets. Singapore, World Scientific, 1991.

2. M u r a t a, T. Petri Nets – Properties, Analysis and Applications. – In: Proc. of IEEE 77, 1989, No 4, 541-580.

3. T r i f o n o v, T., K. G e o r g i e v. GNTicker – a Software Tool for Efficient Interpretation of Generalized Net Models.– Issues in Intuitionistic Fuzzy Sets and Generalized Nets, Vol. **3**, 2006, Wydawnictwo WIT, Warsaw, 71-78.