Optimized Algorithm for Tokens Transfer in Generalized Nets

Dimitar G. Dimitrov[1,2]

[1] Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Sciences, Acad. G. Bonchev Str., bl. 105, 1113 Sofia, Bulgaria

[2] Faculty of Mathematics and Informatics, Sofia University, 5 James Bourchier Str., Sofia, Bulgaria

mitex@gbg.bg, dgdimitrov@fmi.uni-sofia.bg

Abstract

This paper presents an optimized algorithm for token transfer in generalized nets (GNs). Both transition's and GN's functioning are optimized, while keeping the algorithm's original structure and readability. In most cases less predicates are checked, less places are sorted, abstract transition's matrices are optimized, etc. The way of the functioning and the results of the work of a given GN are equal for both the old and the optimized algorithm.

Keywords: Generalized Nets (GNs), algorithm, optimization, token transfer

1 Introduction

Generalized nets (GNs) are introduced in 1982 as an extension of Petri nets and Petri nets modifications and extensions [1]. GNs are an instrument for modelling and optimization of parallel and competitive processes.

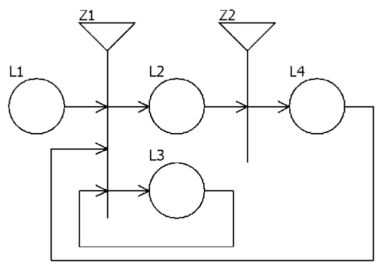
The algorithm for token transfer in GNs [2] is more complex than the corresponding algorithm in Petri nets. Since GNs can be used in large and complex systems, effective algorithm is needed for their functioning. In previous work some optimizations are already made [4], but there is still work to be done.

In this paper both algorithms for token transfer in GNs [2] are optimized, namely Algorithm A for transition's functioning and Algorithm B for GN's functioning. Presented optimizations are intended to reduce the time required

for the algorithms to run. Besides that, some memory saving is also achieved. In previous work [4] only Algorithm A has been improved.
The remainder of this paper is organized as follows. In next section the new optimizations of the algorithm for token transfer are analysed. In Section 3 the new, optimized algorithm is given. Finally, conclusion remarks about current and future work are given.

2 Analysis

Full definition of a generalized net, as well as the previous algorithm for token transfer, can be seen in [2].
In step A*04 of the old version of Algorithm A [2] all predicates in a given row of the predicate index matrix (IM) *r*, which correspond to an empty cell in the IM *R*, are evaluated. If token splitting is not allowed (this fact is determined by the dynamical operator defined over the given GN [2]), the token moves to the highest priority output place, which is permitted for it. Since predicates can be very complex in some transitions, their evaluation should be considered a heavy operation. In the new version evaluation of predicates is performed in order of decreasing priority of the output positions and stops if a given predicate has a value of 1 ("true"). Thus in the middle case about two times less predicates are checked. If token splitting is allowed, the old algorithm remains, i.e. all predicates corresponding to non-zero cells in *R* [2] are evaluated.
The "union" operation used in algorithm B [2] for the construction of the Abstract Transition (AT) requires the creation of huge index matrices for predicates and capacities. These matrices are rectangular block-diagonal, because no place in a GN can be input, respectively output, for more than one transition (Fig. 1.). In the new version of the algorithm, we will keep separate matrices for all transitions which form the AT, instead of merging them into large matrices. This way the number of outputs for a given input is significantly smaller than in a classic AT. Also removing of a transition from AT is easier.



$$\begin{array}{c|ccc} & L_2 & L_3 & L_4 \\ \hline L_1 & * & * & \\ L_3 & * & * & \\ L_4 & * & * & \\ L_2 & & & * \end{array}$$

Fig. 1. A simple GN model and the structure of its Abstract Transition (AT)'s index matrices

Another bottleneck in the old algorithms is the sorting of places. All input and output places of the AT are sorted at each step of GN's functioning. Some places such as empty inputs and full outputs are not involved in the transition's functioning, therefore there's no need to sort them. One way to implement this is to temporarily set the lowest possible priority for such places. Also in some situations the order of the places from given transition is the same in every step so the list of the places can just be merged with the sorted list of places in the AT.

In both algorithm A and algorithm B there is a main loop which operates while a given condition is satisfied (current time is less than a given value). In algorithm A a postcondition loop is used, while in algorithm B a precondition loop is given, i.e. the condition is checked before each iteration. The difference between the postcondition and the precondition loop is that the first can execute at least once, while the latter can finish without performing any iteration. The condition in algorithm B is whether current time is less than $T+t^*$. Before the first iteration there is no possibility for this condition to be dissatisfied, so the loop can be replaced with postcondition one.

3 The Optimized Algorithms

3.1 General Algorithm for a Transition's Functioning at a Given Time Moment t_1 (Algorithm A)

(A01) In each input place are formed two lists:

- a list of all tokens, arranged by priority;
- an empty list.

(A02) An empty IM R, which corresponds to the IM of the predicates r of the given transition, is generated. It is initiated without values. After this, values 0 are assigned (corresponding to value "false") to all of the elements of this IM, which:

- are placed in a row, corresponding to an empty input place, or
- are placed in a column, corresponding to a full output place, or
- are placed in (i, j)-th position, for which $m_{i,j} = 0$, i.e. the current capacity of the arc between i-th input and j-th output place is zero.

(A03) The sorted places are passed sequentially by their priority, starting with the place having the highest priority, which has at least one token and through which no transfer has occurred on the current time-step. For the highest priority token (from the first list) we determine whether it can split or not. This fact is determined by the dynamical operator defined over the given GN. If such an operator is not defined, we will assume that token can split as many times as necessary. After this the predicates corresponding to the relevant row of IM *R* are checked. If the token cannot split, the check finishes with finding the first predicate with truth-value different from 0; in the other case, we must evaluate the truth values of all predicates in the row, for which the elements of *R* are not zeros.

(A04) Depending on the execution of the operator for permission or prohibition of tokens splitting over the net, the token from step **(A03)** will pass either to all permitted for it output places, or to this very place among them, which has the highest priority. If one token cannot not pass through a given transition on this time interval, it is moved to the second list of tokens of the corresponding input place. The tokens, which have entered into the place after the transition activation, are moved into the second list too.

(A05) The values of the characteristic function for the output places (one or more), in which tokens have entered (according to step **(A04)**), are calculated.

(A06) Put values "0" in all rows of *R* for which the respective input place (from where the token goes out on step **(A04)**) is already empty; or in the columns of R which are full in a result of token's transfer on step **(A04)**; or in the IM-places that correspond to arcs between the discussed input place and these output places, for which the arc-capacity becomes to “0” in a result of the token's transfer.

(A07) The current number of tokens in all input places for the current transition decrements with 1 for each token that has go out them at this time step. If the current number of tokens for a given input place is zero, the elements of the corresponding row of IM *R* are assigned value “0”.

(A08) The capacities of all output places, in which a token, determined at step **(A03)**, has entered, decrement with 1. If the maximum number of tokens for a given output place is reached, the elements of the corresponding column of IM *R* are made "0".

(A09) The capacities of all arcs through which a token has passed is decrement with 1. If the capacity of an arc has reached 0, the element from IM *R* that corresponds to this arc is assigned value 0.

(A10) If there are still tokens in the input places that are due to transfer, and there are spare output places, and there are arcs with non-zero capacities, then the algorithm proceeds to step **(A11)**; in the opposite case it proceeds to step **(A13)**.

(A11) The current time is increased with t^0.

(A12) Is current time moment greater than t_1+t_2? If the answer of the question is "no", return to step **(A03)**, otherwise go to **(A13)**.

(A13) Termination of the transition functioning.

3.2 General Algorithm for a GN's Functioning at a Given Time Moment t_T (Algorithm B)

(B01) Put all tokens α for which $\theta_k(\alpha) \leq T$ into their corresponding input places of the net.

(B02) Construct the GN's abstract transition. It is the union of all active GN-transitions at a given time-moment.

(B03) Find all transitions for which the first time-component is exactly equal to the current time moment t_T.

(B04) Evaluate the transitions' types (□) for all transitions, found in step **(B03)**. Use the following method:

- replace the names of all places which participate in the Boolean expression of the transition type with values: 0, if the corresponding place has no tokens at the current moment; 1, otherwise;
- evaluate the truth value of the so obtained Boolean expression.

(B05) Add to AT all transitions from **(B04)** with □ = 1. Take the following steps to add the places to the AT:

- If it is not possible for a place to change its priority during simulation, do the following for each transition to be added:
  - Sort the input and output places, respectively, if they have never been sorted after the start of the simulation.
  - Merge the sorted lists of places, starting with the shortest ones.
- If it is possible for a place to change its priority during simulation:
  - Sort all non-empty input places and all non-full output places in the AT.
  - Sort all non-empty input places and all non-full output places in each transition to be added to the AT.
  - Merge the sorted lists of places, starting with the shortest ones. Append non-sorted sections after the sorted ones.

(B06) Apply algorithm A over the AT.

(B07) Remove from AT all transitions which are inactive at the current time moment. Keep the order of the places from the AT, as well as the order of the places from the transitions.

(B08) Increase current time with t^0.

(B09) Check whether current time moment is greater than $T+t^*$.

(B10) If the answer of the question in step **(B09)** is negative, go to step 2, otherwise – end of GN's functioning.

4 Conclusion

Presented optimizations keep the algorithm's original structure and readability. In most cases less predicates are checked, less places are sorted, abstract transition's matrices are optimized, etc. The way of the functioning and the results of the work of a given GN are equal for both the old and the new algorithm.

The proposed optimizations of Algorithm A and Algorithm B for token transfer in GNs will be implemented in GNTicker [3] – a GN interpreter written in C++. GNTicker is the main component of current software package for generalized nets [5].

Acknowledgement

The author would like to acknowledge the financial support of the Bulgarian National Science Fund under contract DID-02-29.

5 References

[1] Atanassov K. Generalized nets. World Scientific, Singapore, New Jersey, London, 1991, ISBN 978-981-02-0598-0.

[2] Atanassov K. On Generalized nets theory. Prof. Marin Drinov Academic Publishing House, Sofia, 2007, ISBN 978-954-322-237-7.

[3] Trifonov T and K. Georgiev. GNTicker – A software tool for efficient interpretation of generalized net models. *Issues in Intuitionistic Fuzzy Sets and Generalized Nets*, Vol. 3. Warsaw, 2005.

[4] Atanassov, K., V. Tasseva, T. Trifonov, Modification of the algorithm for token transfer in generalized nets. *Cybernetics and Information Technologies*, Vol. 7, 2007, No 1, 62-66

[5] Trifonov T., K. Georgiev, K. Atanassov. Software for modelling with Generalised Nets. *Issues in intuitionistic fuzzy sets and generalized nets*, Vol. 6, 2008, 36-42