Годишник на секция “Информатика”
Съюз на учените в България
Том 5, 2012, 43-52

Annual of “Informatics” Section
Union of Scientists in Bulgaria
Volume 5, 2012, 43-52

ON THE GLOBAL OPERATOR G_6 OVER GENERALIZED NETS

Dimitar G. Dimitrov[1,2]

[1] Faculty of Mathematics and Informatics, Sofia University
5 James Bourchier Str., Sofia, Bulgaria
[2] Institute of Biophysics and Biomedical Engineering
Bulgarian Academy of Sciences
105 Acad G. Bonchev Str., 1113 Sofia, Bulgaria
email: dgdimitrov@fmi.uni-sofia.bg

Abstract: In this paper, a formal definition of the global operator G_6 is given. A new global operator G'_6 is defined. G'_6 extends G_6 and can be used on a wider class of generalized nets. For a given GN E, the results of the work of E and $G'_6(E)$ are the same.
Keywords: Generalized net, Global operator, Formalization, Software implementation, GN Lite.

1. Introduction

Generalized Nets (GNs) are extensions of Petri nets and Petri net modications and extensions [1] [2]. The idea for defining operators over Generalized Nets was introduced in [1], and up to now essentially developed. Each operator maps to a given GN a new GN with specific properties. The operators that can be applied over GNs fall into six categories: global, local, hierarchical, reducing, extending and dynamical operators [3–5]. Global operators transform, according to a definite procedure, an entire given GN or all its components of a given type.

Operator G_6 changes the form and the structure of a given GN E by shrinking each its “linear” subnet into a single transition. We shall denote a linear GN every GN, for which first transition’s outputs are the only inputs of the second one, the second transition’s outputs are the only inputs of the third one, etc. [6]

G_6 operator was introduced in [1] but not formally defined up to now. While the algorithm for applying G_6 (and therefore its software implementation) seems straightforward at first glance, there are many special cases that are not covered by the original definition.

In this paper G_6 operator is formalized and its limitations are discussed. A new global operator, G'_6, is as well introduced. It retains the original idea of G_6, but unlike it, for a given GN E, the results of the work of E and $G'_6(E)$ are the same (on every step of their functioning).

In [7] another global operator, G_{21}, is modified to keep the structure of all removed components. The purpose is to make the operator reversible. In this paper only some properties of the removed components are saved, but with entirely different goal - to preserve the functioning and results of the work of a given GN for each time step.

Proposed here GN operator G'_6 is implemented in GN IDE - a graphical environment for visual editing and simulation of GN models [8]. GN IDE is part of GN Lite - a software package for modeling and simulation with GNs [12]. Currently GN Lite supports GNTCFL and JavaScript as programming languages for predicates and characteristic functions in GN models. The software implementation of G'_6 supports the JavaScript language.

In this paper with $pr_i A$ we shall denote the i-th projection of the n-dimensional set A where $n \in \mathbb{N}$, $1 \leq i \leq n$. A full formal definition of a GN can be found in [2].

2. Formalization of G_6

G_6 operator is not initially formally defined [1]. For the purpose of this paper, as well as for the software implementation of G_6, we shall need a formal definition.

The purpose of G_6 is to replace each "linear" sequence of transitions in a given GN with a single transition. Figure 1 illustrates the effect of G_6 on a sample GN with two linear sequences of transitions.

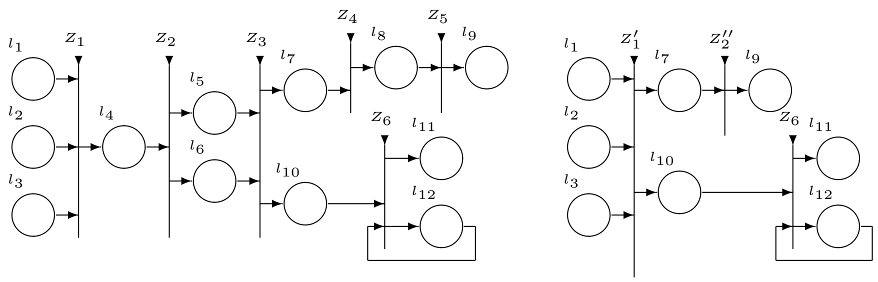

Figure 1: Sample generalized net (left) and the result of applying G_6 on it (right)

We shall need the following definitions:

Definition 1. A *linear sequence of transitions* is the sequnce of transitions $Z_0, ..., Z_{n-1}$ if and only if $pr_2 Z_i = pr_1 Z_{i+1}, 0 \leq i \leq n-2$.

Definition 2. A *linear generalized net* is a GN for which all transitions form a linear sequence.

Operator G_6 can be formally defined as follows, according to the informal description in [1]. G_6 transforms each linear sequence of transitions $Z_0, ..., Z_k$ in E to a transition in the following form:

$$\langle L', L'', t_1, t_2, r, M, \square \rangle$$

The input places for the new transition are all input places of the first transition. The output places are the outputs of the last transition:

$$L' = pr_1 Z_0$$
$$L'' = pr_2 Z_k$$

The first time component of the transition is equal to the first time component of the first transition. The second time component is the sum of the second time components of all transitions belonging to the subnet:

$$t_1 = pr_3 Z_0$$
$$t_2 = \sum_{i=0}^{k} pr_4 Z_i$$

The predicate matrix and the capacity matrix are compositions of old transitions' predicate and capacity index matrices, respectively. Each element in r is a disjunction of conjunctions of all possible paths between the two places, while in M it is the maximum value of all minimum capacities of each possible path. We shall use the following recursive function to define r and M:

$$h(t, p, c_1, c_2, l_i, l_j) = \begin{cases} pr_p(Z_t)_{l_i,l_j} & \text{if } t = 0 \\ c_1\{c_2(h(t-1, p, c_1, c_2, l_i, m), pr_p(Z_t)_{m,l_j})|m \in pr_1 Z_t\} & \text{else} \end{cases}$$

$$r = [pr_1 Z, pr_2 Z, \{h(k, 5, \vee, \wedge, l_i, l_j)\}]$$
$$M = [pr_1 Z, pr_2 Z, \{h(k, 6, \max, \min, l_i, l_j)\}]$$

$$\square = pr_7 Z_1$$

The characteristic function Φ for each output place $l_j \in pr_2 Z$ executes the characteristic functions of all places along all possible paths between the previous place l_i of the token and curren output place l_j. Φ_{l_j} can be defined via the following pseudocode:

begin
    for each possible path $l_i, m_1, ..., m_k, l_j$
        if $W_{l_i,m_1} \wedge W_{m_1,m_2} \wedge ... \wedge W_{m_k,l_j}$
            execute Φ_{m_1}
            ...
            execute Φ_{m_k},
            execute Φ_{l_j}
        end if
    end for
end

The list of all possible paths between l_i and l_j can be generated recursively, using the idea for r and M. $W_{l',l''}$ is the predicate for the arc between l' and l'', taken from the predicate matrix of corresponding transition Z (such that $l' \in pr_1 Z \wedge l'' \in pr_2 Z$). All received characteristic values are remembered in the characteristic history the same way as in E (before applying G_6).

3. A comparison of the functioning of E and $G_6(E)$

The so defined operator G_6 cannot be applied on a given GN E if some of its predicates or characteristic functions depends on a place, removed by the operator. For example, in E a given predicate checks whether l, an internal place of a linear subnet, is non-empty. In $G_6(E)$ place l no longer exists.

If any of the following conditions is not satisfied, E and $G_6(E)$ may not produce the same result:

- Predicates and characteristic functions in a linear subnet must not depend on characteristic values obtained by a characteristic function of a removed place. For example, let token α has initial characteristic equal to 0. The characteristic function for some place l sets it to 1, then some predicate W checks whether the characteristic of α is greater than 0 and the result is $true$. In $G_6(E)$ the combined predicate checks if the characteristic of α is greater than 0, but it is still equal to 0, because the combined characteristic function is not yet executed.
- If a characteristic function Φ_l is executed for a token α, Φ_l should not set characteristics of tokens different than α.
- Transition types of Z_2, ..., Z_k should have the form of disjunctions, i.e. $\vee(l_1, ..., l_n)$, where $l_1, ..., l_n$ are the inputs of the corresponding transition.
- Priorities of all internal (subject to removal) output places of a given transition should be equal. In $G_6(E)$ priorities of individual removed places are lost.
- Capacities of all internal output places of a given transition should be equal.
- Capacities of all removed incoming arcs (outcoming arcs, respectively) of a given transition should be equal.

In the next section a new global operator G'_6 that does not enforce the above restrictions is introduced.

4. New global operator G'_6

The modified version of G_6 compresses all linear paths in a given GN E the same way as G_6, but with one significant difference. For each transition that replaces a linear fragment a loop is added where incoming tokens are collected. If the linear subnet has $k+1$ transitions, tokens pass $k-1$ times through the new place, before they leave the loop and enter some output place. This way the functioning of E in the time is not altered. The characteristic function of the new place combines all of the characteristic functions, predicates, and transition types of all removed components. For each token it imitates the process of passing through different places accross the path. To achieve this, it adds a characteristic named $virtual\_host$ to each token that enters the place. The value of the new characteristic is the nonexisting place, which the token would enter if the operator was not applied. The new loop also simplifies the transformation of predicates and characteristic functions. If a given predicate or function depends on a removed place, it will be modified to refer to the place from the loop and filter the tokens by $virtual\_host$, as described later in this section.

Because of the new loop added, the new version of the operator requires each replaced subnet in E to be the largest possible linear subnet in E, containing the given set of transitions. We shall denote a *maximal linear subnet* every subnet F in E, for which adding any adjacent transition to it makes it non-linear.

Figure 2 shows a GN E with two linear sequences of transitions before and after $G'_6(E)$ is applied.

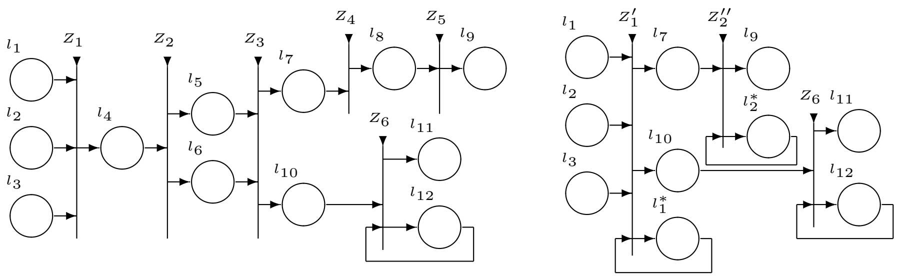

Figure 2: Sample generalized net and the result of applying G'_6 on it

G'_6 transforms the transitions $Z_0, ..., Z_k$ of each maximal linear subnet of a given GN E to a single transition in the following form:

$$\langle L', L'', t_1, t_2, r, *, \square \rangle$$

The input places for the new transitions are all input places of the first transition, as well as a new place l^*:

$$L' = pr_1 Z_0 \cup \{l^*\}$$

The output places are the outputs of the last transition, as well as the new place l^*:

$$L'' = pr_2 Z_k \cup \{l^*\}$$

A new token σ enters place l^*. Its purpose is to save some parameters of the transitions and places removed by G'_6. Its initial characteristic has the following form:

$$x^\sigma = \langle \{\langle l, (pr_4 pr_1 E)(l)\rangle | l \in pr_1 Z_1 \cup ...pr_1 Z_k\},$$
$$pr_6 Z_0 + ... + pr_6 Z_k,$$
$$\{\langle Z, \langle pr_3 Z, pr_4 Z\rangle\rangle | Z \in \{Z_0, ..., Z_k\}\}\rangle,$$

where $pr_4 pr_1 E$ is the capacity function c of E, and $pr_6 Z$ denotes the capacity matrix of a given transition Z. The first component of x^σ keeps the capacities of all internal places in the subnet, the second one is a single index matrix containing the capacities of all arcs, and the third one specifies the time components of the removed transitions.

Token σ enters the net in the first time moment of its functioning.

$$\theta_K(\sigma) = T_1$$
$$b(\sigma) = \infty$$

The first and the second time components of the transition are the same as in G_6:

$$t_1 = pr_3 Z_0$$
$$t_2 = \sum_{i=0}^{k} pr_4 Z_i$$

Since predicates and characteristic functions of E may depend on places removed by G'_6, all predicates and characteristic functions must be modified as follows:

In E	In $G'_6(E)$
getting the place where a given token is situated	getting the value of $virtual\_host$
getting a list of all tokens in place l	getting a list of all tokens in l^* for which $virtual\_host = l$

With $R_{l^*}(f)$ we shall denote a characteristic function or predicate f with substituted function calls as described above. The formal definition of the above substitutions depends on the function language used (e.g. GNTCFL or JavaScript). We shall define $R_{l^*}(\,\Box\,)$ for transition types too. $R_{l^*}(\,\Box\,)$ checks the type of a removed transition:

$$R_{l^*}(l) = \exists\alpha : virtual\_host = l$$

$$R_{l^*}(\vee(\,\Box_1, ..., \Box_n)) = R_{l^*}(\,\Box_1) \vee ... \vee R_{l^*}(\,\Box_n)$$

$$R_{l^*}(\wedge(\,\Box_1, ..., \Box_n)) = R_{l^*}(\,\Box_1) \wedge ... \wedge R_{l^*}(\,\Box_n)$$

where α is a token in l^*.

We shall also define the following function in order to simplify next definitions. It checks whether a given removed transition Z is active at current time moment t using data from σ:

$$active(Z) = pr_1(pr_3\sigma(Z)) \leq t \leq pr_1(pr_3\sigma(Z)) + pr_2(pr_3\sigma(Z))$$

The predicate matrix has the following form:

$$r = \begin{array}{c|cccc} & l''_1 & \dots & l''_m & l^* \\ \hline l'_1 & false & \dots & false & W_{1,*} \\ \vdots & \vdots & & \vdots & \vdots \\ l'_n & false & \dots & false & W_{n,*} \\ l^* & W_{*,1} & \dots & W_{*,m} & true \end{array}$$

where:

$$W_{i,*} = active(Z_0) \wedge pr_7 Z_0 = true \wedge (W'_{l'_i,m_0} \vee ... \vee W'_{l'_i,m_p})$$

for every $1 \leq i \leq n$, where $pr_2 Z_0 = \{m_0, ..., m_p\}$ are the outputs of the first transition, ordered by priority in descending order, i.e. $\pi_L(m_0) \geq ... \geq \pi_L(m_p)$, and

$$W'_{l',l''} = pr_1 x^{\sigma}(l'') > 0 \wedge pr_2 x^{\sigma}{}_{l',l''} > 0 \wedge R_{l^*}(pr_5 Z_0)_{l',l''};$$

$$W_{*,j} = active(Z_k) \wedge virtual\_host \in pr_1 Z_k \wedge R_{l^*}(pr_7 Z_k) = true$$

$$\wedge pr_2 x^{\sigma}{}_{virtual\_host,l''_j} > 0 \wedge R_{l^*}(pr_5 Z_k)_{virtual\_host,l''_j}$$

for every $1 \leq j \leq m$.

Predicate evaluation intentionally is placed at the end of the corresponding expression. In optimized versions of the algorithm for token transfer in GNs [9] [2] predicates are not checked if corresponding output places are full, capacity of corresponding arcs are zeroes, etc. The software implementation of the conjunction operator in many programming languages is optimized so if the left argument is equal to false, the right argument is not evaluated.

The transition's type is as follows:

$$\square = pr_7 Z_0 \vee l^*,$$

where $pr_7 Z_0$ is the type of the first transition (Z_0).

The new transition has the highest priority of all transitions in the linear subnet:

$$\pi_A(Z) = \max(\pi_A(Z_0), ..., \pi_A(Z_k)) + 1$$

The new place l^* has the highest priority among all places in the whole net E:

$$\pi_L(l^*) = \max\{\pi_L(l) | l \in L \setminus \{l^*\}\} + 1,$$

where

$$L = pr_1 pr_1 pr_1 E \cup pr_2 pr_1 pr_1 E$$

are all places in E.

The capacity of l^* is infinity, i.e. $c(l^*) = \infty$. This is because token movement decisions for the removed places are based on the information saved in σ. Same aplies for transition times. They are not modified by θ_1^Z and θ_2^Z.

The characteristic function $\Phi_{l'_i}$, where $1 \leq i \leq n$, extends the previous $R_{l^*}(\Phi_{l'_i})$. In addition to the old behavior, the new function also decreases the capacities of the corresponding arcs and places in σ.

$\Phi_{l''_j}$, where $1 \leq j \leq m$, extends the previous characteristic function for l''_j. It removes the $virtual\_host$ characteristic from each token that enters the corresponding place. This is done before the original functionality of $\Phi_{l''_j}$ (again modified by R_{l^*}).

Φ_{l^*} imitates the functioning of the removed subnet. We shall define it via the following pseudocode (α is current token for which Φ_{l^*} is executed):

begin
  if $virtual\_host$ is not set — i.e. α has just entered l^*
    $prev \leftarrow$ corresponding removed place for which predicate W' is evaluated as $true$
    $virtual\_host \leftarrow prev$
  else
    $Z \leftarrow$ the only Z for which $virtual\_host \in pr_1 Z$
    if $active(Z) \wedge R_{l^*}(pr_7 Z) = true$
      $outputs \leftarrow pr_2 Z$, sorted by priority in descending order
      if $\exists l \in outputs : (pr_2 x^\sigma)_{virtual\_host,l} > 0 \wedge pr_1 x^\sigma(l) > 0 \wedge R_{l^*}(pr_5 Z)_{virtual\_host,l}$
        $prev \leftarrow virtual\_host$
        $virtual\_host \leftarrow l$
      end if
    end if
  end if
  $pr_1 x^\sigma(virtual\_host) \leftarrow pr_1 x^\sigma(virtual\_host) - 1$ — decreasing capacity of $virtual\_host$ in σ.
  $(pr_2 x^\sigma)_{prev,virtual\_host} \leftarrow (pr_2 x^\sigma)_{prev,virtual\_host} - 1$ — decreasing corresponding arc capacity.
  execute $R_{l^*}(\Phi_{virtual\_host})$
  $pr_1(pr_3\sigma(Z)) \leftarrow pr_6 pr_1 E(pr_1(pr_3\sigma(Z)))$ — updating t_1 of Z in σ using θ_1
  $pr_2(pr_3\sigma(Z)) \leftarrow pr_7 pr_1 E(pr_2(pr_3\sigma(Z)))$ — updating t_2 of Z in σ using θ_2
end

Operator G'_6 also transforms predicates of all retained transitions $\{Z|Z \in pr_1 pr_1 E \setminus \{Z_0, ..., Z_k\}\}$ and the characteristic function for all retained places by applying operator R_{l^*}, described earlier in this section. This ensures that no predicate or function in $G'_6(E)$ will function properly after removing the unnecessary places from the net.

For significantly wider class of generalized nets for which token splitting and merging is not allowed, i.e. dynamical operators $DD(2,1)$ and $DD(2,4)$ [2] is not be defined for them, every given GN E and $G'_6(E)$ function in the same way and the result of their work is the same. Since GNs are Turing-complete, it is not always possible for G'_6 to retain the functioning of a given GN. For example, if some function, no matter if it knows about any new σ token, iterates over all tokens in all input or output places of the first or last transition in some linear subnet, it may eventually reach σ and modify it, making the result of $G'_6(E)$'s functioning unpredictable.

5. Conclusion

The so-defined new global operator simplifies a given generalized net E while not altering its functioning and the results of its work. G_6 and G'_6, as well as the other GN operators, can be used for model refactoring.

G_6 is important for checking properties (correctness, equivalence, etc.) of procedural [10] and object-oriented programs [11].

Future work on the topic will include:

- Extending G'_6 to support token splitting and merging.
- Formalization and extensions of other GN operators.

Acknowledgements

This work is supported by the National Science Fund Grant DID 02/29 “Modelling of Processes with Fixed Development Rules (ModProFix)”.

References

[1] Atanassov, K. *Generalized Nets*, World Scientific, Singapore, 1991.

[2] Atanassov, K. *On Generalized Nets Theory*, Prof. M. Drinov Academic Publishing House, Sofia, 2007.

[3] Atanassov K. Operator aspect of the theory of generalized nets. *AMSE Review*, Vol. 4, 1987, No. 4, 27–30.

[4] Atanassov, K. Global operators defined on the set of generalized nets. *AMSE Review*, Vol. 4, 1987, No. 4, 31–46.

[5] Atanassov, K., E. Sotirova. On Global Operator G_{21} Defined over Generalized Nets. *Cybernetics and Information Technologies*, Vol. 4, 2004, No. 2, 30–40.

[6] Atanassov, K. *Generalized Nets and Systems Theory*, Prof. M. Drinov Academic Publishing House, Sofia, 1997.

[7] Sotirova, E. On a modification of the global operator G_{21} over generalized nets. *Proc. Jangjeon Math. Soc.*, Vol. 9, 2006, No. 1, 65–78.

[8] Dimitrov, D. G. GN IDE - A Software Tool for Simulation with Generalized Nets, *Proc. of Tenth Int. Workshop on Generalized Nets*, Sofia, 5 December 2009, 70–75.

[9] Dimitrov, D. Optimized algorithm for token transfer in generalized nets. *Recent Advances in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics, Vol. I: Foundations*, 2010, 63–68.

[10] Todorova, M. Verification of Procedural Programs via Building their Generalized Nets Models. *Proceedings of the 41. Spring Conference of the Union of Bulgarian Mathematicians, Mathematics and Education in Mathematics*, 2012, 259–265.

[11] Todorova, M. Construction of Correct Object-Orientated Programs via Building their Generalized Nets Models, *Annual of "Informatics" Section Union of Scientists in Bulgaria*, Vol. 4, 2011, 1–28.

[12] 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.