Issues in IFSs and GNs, Vol. 12, 2015/2016, 80–92

Implementation of the Reducing Operators over Generalized Nets in GN IDE

Nora Angelova[1], **Dafina Zoteva**[2]

Dept. of Bioinformatics and Mathematical Modelling
Institute of Biophysics and Biomedical Engineering,
Bulgarian Academy of Sciences
105 Acad. G. Bonchev Str., 1113 Sofia, Bulgaria
e-mais: metida.su@gmail.com, dafy.zoteva@gmail.com

Abstract: This paper presents the software implementation of the reducing operators over generalized nets (GNs). They are integrated in GN IDE (GN Integrated Development Environment) – a software tool for creating and simulating GN models. This allows the user to optimise the performance of a GN and to implement different test scenarios over the model.
Keywords and phrases: Generalized Nets, Reducing operators, GN IDE.
2000 Mathematics Subject Classificatio: 68Q85.

1 Introduction

Generalized nets (GNs) are a means of modelling parallel and concurrent processes [2, 3].

GNs models, that are being created, are repeatedly changed during their design process. The need of optimizing (or simplifying) the model is often found after completing the simulation of the designed process. The reducing operators can be used at that point when certain GN components are found to be redundant for the model and thus can be omitted.

Two new features related to the reducing operators are integrated in GN IDE. They will be presented here. The first one allows the reduction of components from an already created GN without any loss of information so that it

can be possible for user to revert the action of the applied reducing operators. The second one allows the identification of classes of reduced GNs when they are imported in GN IDE.

The implementation of these two features make it possible for the user to explore different test scenarios with the creation of a single GN model. Besides that reducing certain GN components leads to optimisation of the transitions performance and hence of the GN performance as a whole. It is possible because of the reduction of the number of operations performed by the algorithm of transition's functioning where merging of tokens is allowed [1].

The next sections of the current paper briefly discuss the definitions of the GNs, GN IDE, reducing operators and their software implementation.

2 GNs, Reduced GNs, GN IDE

2.1 Generalized net definition

GNs are a universal tool for describing structured and reusable models of complex systems usually involved in parallel, simultaneous activities. The modelled process is separated into particular events by an expert who is well acquainted with it. A GN transition has to be assigned to each of these events [3].

Formally, every transition in a GN is described by a seven-tuple:

$$Z = \langle L', L'', t_1, t_2, r, M, \Box \rangle,$$

where:

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

(b) t_1 is the current time-moment of the transition's firing;

(c) t_2 is the current value of the duration of its active state;

(d) r is the transition's *condition* that determines which tokens will pass (or *transfer*) from the transition's input to its output places; it has the form of an Index Matrix (IM; see [4]):

$$r = \begin{array}{c|c} & l''_1 \ \ldots \ l''_j \ \ldots \ l''_n \\ \hline l'_1 & \\ \vdots & r_{i,j} \\ l'_m & \end{array};$$

$r_{i,j}$ is the predicate that corresponds to the i-th input and j-th output place $(1 \leq i \leq m, 1 \leq j \leq n)$. When its truth value is "*true*", a token from the i-th input place is transferred to the j-th output place; otherwise, this is not possible;

(e) M is an IM of the capacities $m_{i,j}$ of transition's arcs, where $m_{i,j} \geq 0$ is a natural number;

(f) $\Box$ is the transition type. It is an object of a form similar to a Boolean expression. It contains as variables the symbols that serve as labels for a transition's input places. The expression is built upon these variables and the Boolean connectives $\wedge$ and $\vee$. The transition can become active only when the value of its type (calculated as a Boolean expression) is "*true*".

The ordered four-tuple

$$E = \langle\langle A, \pi_A, \pi_L, c, f, \theta_1, \theta_2\rangle, \langle K, \pi_K, \theta_K\rangle, \langle T, t^o, t^*\rangle, \langle X, \Phi, b\rangle\rangle$$

is called a GN if:

(a) A is a set of transitions;

(b) π_A is a function that assigns priorities to the transitions, i.e., $\pi_A : A \to N$, where $N = \{0, 1, 2, \dots\} \cup \{\infty\}$;

(c) π_L is a function that assigns priorities to the places, i.e., $\pi_L : L \to N$, where $L = pr_1 A \cup pr_2 A$, and $pr_i X$ is the i-th projection of the n-dimensional set, where $n \in N, n \geq 1$ and $1 \leq i \leq n$;

(d) c is a function that assigns capacities to the places, i.e., $c : L \to N$;

(e) f is a function that calculates the truth values of the predicates of the transition's conditions (for the GN described here, let the function f have a value from the set $\{0, 1\}$);

(f) θ_1 is a function which indicates the next time-moment when a certain transition Z can be activated, that is, $\theta_1(t) = t'$, where $pr_3 Z = t, t' \in [T, T + t^*]$ and $t \leq t'$. The value of this function is calculated at the moment when the transition stops functioning;

(g) θ_2 is a function which gives the duration of the active state of a certain transition Z, i. e., $\theta_2(t) = t'$, where $pr_4 Z = t \in [T, T + t^*]$ and $t' \geq 0$. The value of this function is calculated at the moment when the transition starts functioning;

(h) K is the set of the GN's tokens.

(i) π_K is a function that assigns priorities to the tokens, that is, $\pi_K : K \to N$;

(j) θ_K is a function that evaluates the time-moment when a given token can enter the net, that is, $\theta_K(\alpha) = t$, where $\alpha \in K$ and $t \in [T, T + t^*]$;

(k) T is the time-moment when the GN starts functioning. This moment is determined with respect to a fixed (global) time-scale;

(l) t^o is an elementary time-step, related to the fixed (global) time-scale;

(m) t^* is the duration of the GN's functioning;

(n) X is the set of all initial characteristics which the tokens can receive when it enters the net;

(o) 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.

(p) b is a function that returns the maximum number of characteristics a given token can obtain, that is, $b : K \rightarrow N$.

A given GN may not have some of the above components. In these cases, any redundant component will be omitted. The GNs of this kind form a special class of GNs called reduced GNs.

The formal definition of the reduced GNs, as it is stated in [2, 3], is shown in the next subsection.

2.2 Definition of reduced GNs and reducing operators

Let Σ be the class of all GNs. Let

$$\Omega = \{A, \pi_A, \pi_L, c, f, \theta_1, \theta_2, K, \pi_K, \theta_K, T, t^o, t^*, X, \Phi, b\} \cup \{A_i | 1 \leq i \leq 7\},$$

where $A_i = pr_i A (1 \leq i \leq 7)$, i.e. $A_i \in \{L', L'', t_1, t_2, r, M, \square\}$ be the set of all GN's components.

Let $Y \in \Omega$ then Σ^Y shall be the class of those GNs without the Y component [2, 3].

The graphical structure of a GN (the component A), the sets of input and output places in its transitions (the components A_1 and A_2 respectively) and the set of all tokens (the component K) are required in the definition of a GN. Therefore they cannot be omitted or reduced.

$$\Sigma^A = \Sigma^{A_1} = \Sigma^{A_2} = \Sigma^K = \emptyset.$$

If $Y_1, Y_2, \ldots, Y_s \in \Omega$ for $s \geq 1$ then $\Sigma^{Y_1, Y_2, \ldots, Y_s}$ is called $(Y_1, Y_2, \ldots, Y_s)$ – class of reduced GNs.

The reducing operators defined over GNs bring together an ordinary GN and its reduced ones. They are closely connected to the corresponding classes of reduced GNs [2, 3].

If Y is a component of a given GN E then the operator R_Y reduces E to a GN without the component Y, $R_Y(E) \in \Sigma^Y$.

The reducing operators R_Y for $Y \in \Omega$ can be represented through one universal operator $R : (\forall E \in \Sigma)(\forall Y \in \Omega)(R(E, Y) = R_Y(E))$.

For $Y_1, Y_2 \in \Omega : R(R(E, Y_1), Y_2) = R(R(E, Y_2), Y_1), \forall E \in \Sigma$.

The next subsection will introduce shortly GN IDE, the tool which these reducing operators are integrated into.

2.3 Definition of GN IDE

GN IDE is a software tool for creating GN models and running simulations for ones. The simulation can be run and completed altogether or paused and resumed again so that the process can be observed in steps. The tool is written in Java and it is platform independent [5,6]. Each component of the GN models is described like an object in the code with some properties which characterizes it. The structure of the GN model in GN IDE is defined in an XGN file (GN XML).

Each model can be created in the GN IDE or directly with the XML code.

So far a GN IDE model includes description of it's transitions, places, arcs, tokens, matrices (predicates), characteristic functions and data.

3 Integration of the Reducing Operators in GN IDE

3.1 Requirements for the implementation

Currently the description of a GN model is stored in a GN XML file (XGN). The structure of a valid GN model is strictly defined in a GN XML schema file that describes the legal building blocks for the model and their specific order. A detailed description of the GN XML schema is presented in [5]. It has one root element gn which represents the GN itself and four nested elements – transitions (with nested their input and output places, connecting arcs and predicate matrix), places, tokens and functions (characteristic functions and predicates).

The integration of the reducing operators in GN IDE involves a change in the GN XML schema. The following requirements are placed on the implementation:

- compatibility of the previously created GN models with the new GN XML schema so that their simulation and reduction should be possible after the changes;
- an ability to revert the effect of applied reducing operators and restoring the GN original state, i.e. the reducing operators should not change the models.

Keeping them in mind the reduced components are not omitted when the model is stored in a GN XML file but instead only marked as such. They can still hold their values (or some default ones) but as they are marked as reduced these values cannot and will not be used in the process of a simulation.

A new element called $reducedComponents$ is added to the GN XML schema in order to hold the necessary information for the reduced components. It is a child of the root element gn. It is placed right after the $functions$ element and has the following form:

```
<xsd:element name="reducedComponents" minOccurs="0" maxOccurs="1">
    <xsd:complexType>
        <xsd:sequence maxOccurs="1">
            <xsd:element name="T1" minOccurs="0" maxOccurs="1">
                <xsd:complexType/>
            </xsd:element>
            <xsd:element name="T2" minOccurs="0" maxOccurs="1">
                <xsd:complexType/>
            </xsd:element>
            <xsd:element name="M" minOccurs="0" maxOccurs="1">
                <xsd:complexType/>
            </xsd:element>
            <xsd:element name="TRANSITION_TYPE" minOccurs="0" maxOccurs="1">
                <xsd:complexType/>
            </xsd:element>
```

Figure 1: GN XML schema

Figure 1 shows only a part of the definition of this new element $reducedComponents$. The rest of the components that can be reduced are described in the same way.

The definition of the $reducedComponents$ element consists of a sequence of child elements. Each of them is an empty element described by its name only. The names of the child elements are unique and correspond to the GN components that can be reduced. The child elements are not required. If missing, it is assumed that they are not reduced.

Here is the list of all GN's components that can be reduced, the way they are used in XGN files: *T1, T2, M, TRANSITION_TYPE, TRANSITIONS_PRIORITIES, PLACES_PRIORITIES, PLACES_CAPACITIES, TOKENS_PRIORITIES, ThK, T, T0, TStar, INITITIAL_CHARACTERISTICS, CHARACTERIC_FUNCTION, CHARACTERISTICS_NUMBER*.

An example for the $reducedComponents$ element, part of a XGN file is shown here:

```
<reducedComponents>
        <M/>
        <PLACES_PRIORITIES/>
        <PLACES_CAPACITIES/>
</reducedComponents>
```

The $reducedComponents$ element itself is not required in the GN XML schema either. This condition ensures the compatibility between the previously created GN models and the new GN XML schema. It is assumed by default that they are not reduced.

In addition the software implementation does not allow the reduction of transitions' input and output places (A_1, A_2), tokens (K) and the predicates that determine tokens transfer between input and output places of the transitions (r).

3.2 Code implementation

The list of the reduced components in the GN IDE code is stored as an array of bits. If a certain component is reduced the value of its corresponding bit is set to 1, otherwise it is 0. The implementation of the reducing operators is carried out in a new independent layer, where only changes in the bits values take place. Later the presence of reduced components is considered by the

algorithms of transitions and GN functioning. If possible certain steps of the algorithms can be skipped during the simulation process. Being independent from the rest of the software product, the reduced operators can be applied in addition to other operators, relations, etc. This also makes it possible to revert the effect of an applied reducing operator without any loss of information for the GN model.

Applying and removing reducing operators over a GN model can be performed in two ways: by direct modification of a XGN file or by using the GN IDE user interface.

Applying and removing reducing operators by direct modification of a XGN file include the following steps:

- open the XGN file with the GN model that needs to be reduced (or changed) in any text editor;
- add the $reducedComponents$ element in the file, if it is not there yet (after the $functions$ element);
- for each component that should be reduced, add a corresponding child element to the sequence in the parent one ($reducedComponents$). Use the name of the component in the tag of the element.
- for those of the components that should not be reduced but can be found among the child elements in the parent $reducedComponents$, delete the whole element;
- save the XGN file and load it in GN IDE.

GN IDE user interface allows applying and removing reducing operators over a loaded GN model. These options are accessible from *Apply Operator* menu item in *GN* menu on the top menu bar. A check list with the available reducing operators is shown when this option is selected.

If a certain set of GN components have already been reduced from the current GN, when the list of GN components (Figure 2) is loaded the reduced ones are checked. The list of reduced components is extracted from the corresponding XGN file.

Those of the components that are to be reduced have to be marked as checked. This action correspond to applying a reducing operator over the current GN. Removing an existing check mark corresponds to removing a reducing operator.

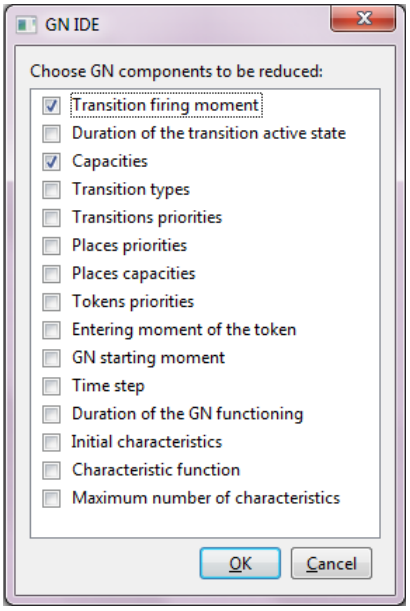

Figure 2: Components list

The dialogue box has two buttons: *OK* and *Cancel*. When the *OK* button is pressed the information for the selected components is stored in the current GN object, then the dialogue box is closed. When the *Cancel* button is pressed all the changes are disregarded, i.e. the current GN object is not changed and then the dialogue box is closed. The *Save* command from *File* menu allows users to transfer the changes from the GN object into the corresponding XGN file.

GN IDE allows users to revert the effect of the reducing operators all at once. This action is accessible again from *Apply Operator* menu item in *GN* menu. It removes the $reducedComponents$ element from the XGN file that corresponds to the current GN. Later if a simulation process is started it uses the original data in the GN components. Reversing the action of the reducing operator will restore the original state of the GN model without any loss of information.

4 Identification of classes of reduced GNs

Identification of the class of reduced GNs, which a GN model belongs to, consists in checking which of the GN components are reduced, if there are any at all.

An important note is that a GN is considered reduced if and only if a reducing operator is applied directly on it. Using the default values for any of the components is not considered a reduction as they may be an accidental result of the current data of the simulation and can not be accepted for the net.

Therefore, once again, all GNs created so far are considered a non-reduced GNs. This state can be changed by using any of the means described above.

Identification of the class of reduced GNs, which a GN model belongs to, can be performed by direct check of the corresponding XGN file or by GN IDE user interface.

When the corresponding XGN file is used directly the act of the class identification actually comes down to establishing the presence of the element $reducedComponents$. The list of the reduced components can be extracted from the sequence of its child elements.

When a certain GN is loaded in GN IDE users are allowed to check if it is reduced. This option is accessible from *Apply Operator* menu item in *GN* menu. If the current GN is reduced, a proper message with a list of reduced components is shown, so the exact class of reduced GNs which the current GN belongs to is determined. The information is extracted from the corresponding XGN file.

A few changes in GN IDE user interface have been made to reflect the results of applying a reducing operator and to facilitate the identification of a reduced GN.

GN IDE user interface features a *Tree view* [5, 6] in its main window. This view shows the hierarchical structure of the GN model as it is stored in the XGN file, starting from the root down to its child nodes. If the current GN is reduced an additional node called *Reduced Components* is shown at the bottom of the *Tree view*. This node can be expanded. Its child nodes are all the GN reduced components. If the current GN is not reduced no such node is included in the *Tree view*.

Figure 3 shows a non-reduced GN loaded in GN IDE. There are no reduced components in the current GN and since the *Reduced Components* node is not required, it is invisible for users.

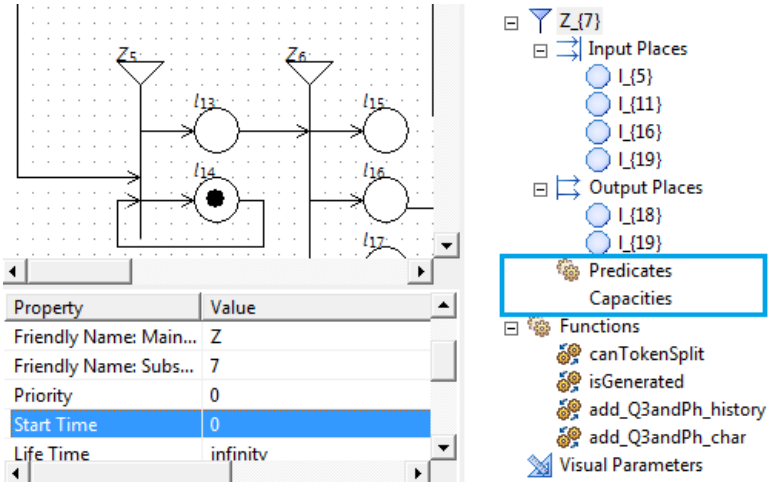

Figure 3: No reduced components

On the other hand this node contains all the GN's reduced components when there are any. So its very presence in the *Tree view* allows users to determine at first glance that the current GN is reduced and the exact class of reduced GNs it belongs to.

Let operators for reducing the index matrix with the transition arcs' capacities and the transition's firing moment are applied to the current GN loaded in GN IDE (see Figure 4).

As it is shown in Figure 4 the *Reduced Components* node is visible and contains two child nodes for each of components that are reduced in the current or any previous simulation of the model.

Besides, in Figure 3 and Figure 4 again, selecting a transition Z_7 in the *Tree view* leads to loading the properties of the corresponding object in the *Property view* at the bottom of GN IDE main window [6]. All but the information for the reduced components is displayed. It is invisible for the user.

The user interface provides an updated view which does not include any information that will not be used in the simulation of a reduced GN.

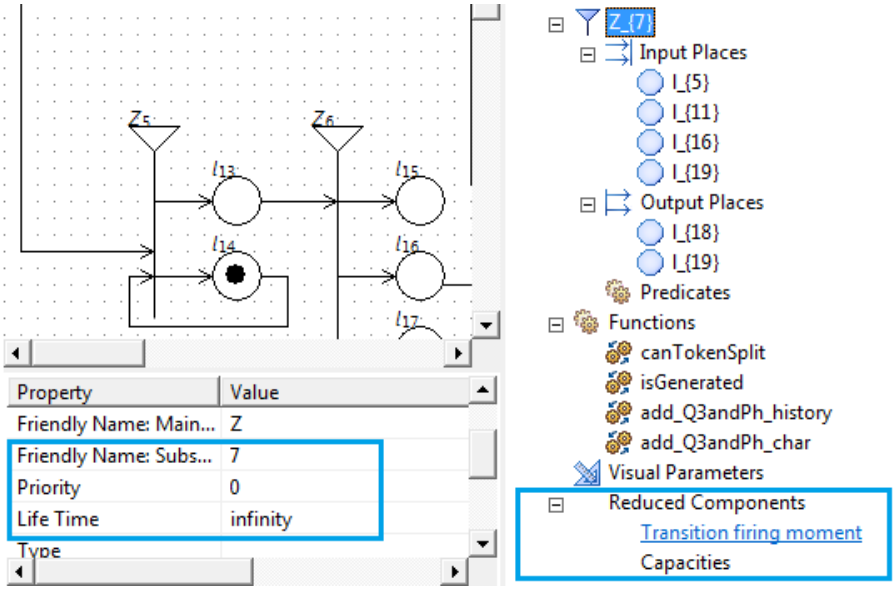

Figure 4: Reduced components

5 Conclusion

GN IDE modelling power can be increased by the consistent implementation of the operators defined over GNs. The current paper describes the starting efforts in this direction by implementing the reducing operators. These operators can affect the algorithm of a GN functioning and thus the results of its work. Their introduction into the simulation tool GN IDE enables the examination of all test scenarios for a GN model. This way the most appropriate model for the simulated process can be chosen.

The further work on this matter will continue with the programming implementation of the operations union, composition, difference and intersection defined for transitions and GNs, followed by the implementation of the rest of the operators over GNs.

References

[1] Andonov, V., N. Angelova. Modifications of the algorithms for transition functioning in GNs, GNCP, IFGNCP1 and IFGNCP3 when merging of

tokens is permitted. *Imprecision and Uncertainty in Information Representation and Processing* (P. Angelov, S. Sotirov, Eds.), Springer, Cham, 2016, 275–288.

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

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

[4] Atanassov, K. *Index Matrices: Towards an Augmented Matrix Calculus*, Springer, Cham, 2014.

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

[6] Angelova N., M. Todorova, K. Atanassov, GN IDE: Implementation, Improvements and Algorithms, *Comptes rendus de l'Academie bulgare des Sciences*, Vol. 69, 2016, No. 4, 411–420.