Applying Floyd's Inductive Assertions Method for Verification of Generalized Net Models Without Temporal Components

Magdalina Todorova[1], Nora Angelova[2]
Faculty of Mathematics and Informatics
Sofia University "St. Kl. Ohridski"
Sofia, Bulgaria

***Abstract*—Generalized Nets are extensions of Petri Nets. They are a suitable tool for describing real sequential and parallel processes in different areas. The implementation of correct Generalized Nets models is a task of great importance for the creation of a number of applications such as transportation management, e-business, medical systems, telephone networks, etc. The cost of an error in the models of some of these applications can be very high. The implementation of models of similar applications has to use formal approaches to prove that the developed models are correct. A foundation stone of software verification, which is suitable for verification of Generalized Nets models with transitions without temporal component, is Floyd's inductive assertion method. This article presents a modification of Floyd's inductive assertion method for verification of flowcharts, which allows Generalized Nets without temporal component to be verified. Using an illustrative example, we show that the offered adaptation is appropriate for the purpose of training university students in the Informatics and Computer Sciences in formal methods of verification.**

***Keywords*—Floyd's inductive assertions method; generalized nets; verification; formal methods; education**

I. Introduction

Generalized Nets (GNs) [1, 2] are a means of modeling sequential and parallel processes in a variety of areas, including medicine, industry, transport, software protection, etc. They were introduced in 1982 by Krassimir Atanassov as a further extension of the standard Petri Nets (PNs) and their modifications and extensions. GNs are defined in a way that is fundamentally different from the ways of defining Regular PNs, E-nets, Time PNs, Colored PNs, Self-modifying PNs, Stochastic PNs, Predicate-transition nets, and other PNs. In the 1980s, it was proved that the functioning and the results of the work of each of these types of nets can be described by a GN. Moreover, it was proved that for each of the classes of standard or extended PNs, there exists a GN that is universal for this class, i.e., it represents the functioning and the results of the work of each of the elements of the respective class of nets. In the following years, similar results were published for the Super nets, Numerical PNs, Fuzzy PNs, and others PNs.

Automatic tools for execution of GNs [3–10] have been developed and are currently in process of improvement.

In parallel with the scientific research related to GN, a great number of GN models have been developed, which simulate real-life processes. Designing such GN models is useful as researching their characteristics allows for focusing the attention of the real-life system developers on the most important (from the point of view of performance quality) elements, as well as to eliminate the unnecessary details when realizing the real-life systems. After developing a GN model, which presents the behavior of a real-life system, research should be conducted to discover how adequate is the developed model, as evaluated against the criteria for the respective real-life system.

Checking the model adequacy is done in two steps: model verification (if it meets the requirements), and model validation (if the requirements posed to the model are adequate to the real-life system). For some models, such a check is obligatory to be done.

This article presents a method of formal verification of GNs, which are without temporal components. The method is a modification of Floyd's inductive assertion method for verification of flowcharts.

The rest of the paper is organized as follows. Part II presents the definitions of a GN, of its main component – the transition, as well as of the GN loop. Part III is dedicated to an adaptation of Floyd's method of verification of flowcharts for verification of GNs with transitions without temporal components. As a result, methods for proving the partial correctness and the termination of such GNs are proposed. In Part IV, these methods are illustrated by a simple example. Applying Floyd's method for verification of practically applicable GN models is a subject of another paper due to volume constrains. Part V of the paper provides comments on applying the presented method for GN verification and ideas for further research in the field.

II. Generalized Nets

GNs are defined in a way that [1, 2] is fundamentally different from the ways of defining the other types of PNs.

Definition 1 (*Transition*). Every transition is described by a seven- tuple:

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

where:

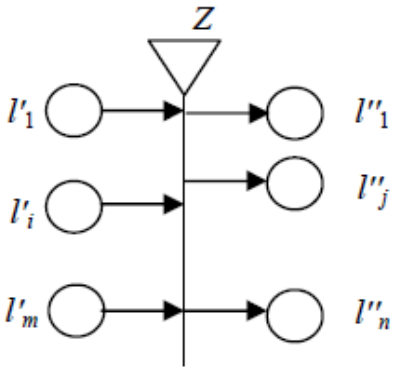

Fig. 1. GN Transition

- L' and L'' are finite, non-empty sets of places (the transition's input and output places, respectively); for the transition Z in Fig. 1 these are: $L' = \{l'_1, l'_2, ..., l'_m\}$ and $L'' = \{l''_1, l''_2, ..., l''_n\}$;
- t_1 is the current time-moment of the transition's firing;
- t_2 is the current value of the duration of its active state;
- r is the transition's condition, determining which tokens will transfer from the transition's inputs to its output places. The parameter r has the form of an Index Matrix (IM) [1, 11]:

r	l''_1	...	l''_j	...	l''_n
l'_1					
⋮			$r_{i,j}$		
l'_m			($r_{i,j}$ – predicate)		
			($1 \le i \le m, 1 \le j \le n$)		

where $r_{i,j}$ is the predicate which gives the condition for transfer from the i-th input place to the j-th output place. When $r_{i,j}$ has truth-value "*true*", then a token from the i-th input place can be transferred to the j-th output place; otherwise, this is impossible;

- M is an IM of the capacities of transition's arcs:

$M =$

	l''_1	...	l''_j	...	l''_n
l'_1					
⋮			$m_{i,j}$		
l'_i			($m_{i,j} \ge 0$ – natural number or ∞)		
⋮			($1 \le i \le m, 1 \le j \le n$)		
l'_m					

- □ is called transition type. It is an object having a form similar to a Boolean expression. It may contain as variables the symbols that serve as labels for transition's input places. It is an expression consisting of variables and the Boolean connectives $\wedge$ and $\vee$ determining the following conditions:

  $\wedge\left(l_{i_1}, l_{i_2}, ..., l_{i_u}\right)$ – every place $l_{i_1}, l_{i_2}, ..., l_{i_u}$ must contain at least one token,

  $\vee\left(l_{i_1}, l_{i_2}, ..., l_{i_u}\right)$ – there must be at least one token in all places $l_{i_1}, l_{i_2}, ..., l_{i_u}$, where $\{l_{i_1}, l_{i_2}, ..., l_{i_u}\} \subset L'$. When the value of a type □ (calculated as a Boolean expression) is "*true*", the transition can become active, otherwise it cannot.

Definition 2 (*Generalized Net*) [1, 2]: 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 Generalized Net if:

- A is a set of transitions (Definition 1);
- π_A is a function giving the priorities of the transitions, i.e., $\pi_A : A \to N$, where N is the set of natural numbers;
- π_L is a function giving the priorities of the places, i.e., $\pi_L : L \to N$, where L = $pr_1 A \bigcup pr_2 A$. Naturally, L is the set of all GN-places;
- c is a function giving the capacities of the places, i.e., $c : L \to N$;
- f is a function that calculates the truth values of the predicates of the transition's conditions. For the (ordinary) GNs, described in this section, the function f obtains values "false" or "true", or values from set {0, 1}. If P is the set of the predicates used in a given model, then f can be defined as $f : P \to \{0, 1\}$;
- θ_1 is a function giving the next time-moment for which a given transition Z can be activated, i.e., $\theta_1(t) = t'$, where $pr_3\, Z = t$, $t' \in [T, T + t^*]$ and $t \le t'$. The value of this function is calculated at the moment when the transition terminates its functioning;
- θ_2 is a function giving the duration of the active state of a given transition Z, i.e., $\theta_2(t) = t'$, where $\mathrm{pr}_4 Z = t \in [T, T + t^*]$ and $t' \ge 0$. The value of this function is calculated at the moment when the transition starts functioning;
- K is the set of the GN's tokens;
- π_K is a function giving the priorities of the tokens, i.e., $\pi_K : K \to N$;

- θ_K is a function giving the time-moment when a given token can enter the net, i.e., $\theta_K(\alpha) = t$, where $\alpha \in K$ and $t \in [T, T + t^*]$;
- T is the time-moment when the GN starts functioning. This moment is determined with respect to a fixed (global) time-scale;
- t^o is an elementary time-step, related to the fixed (global) time-scale;
- t^* is the duration of the GN functioning;
- X is a function which assigns initial characteristics to each token when it enters input places of the net;
- Φ is the characteristic function which assigns new characteristics to each token when it transfers from an input to an output place of a given transition;
- b is a function giving the maximum number of characteristics a given token can receive, i.e.,

$$b : K \rightarrow N.$$

It can be concluded that similarities between PNs and GNs exist, however, there are also differences. The GN transitions have a more complex structure of that of the PN ones. It must be noted that the GN transition contains: an index matrix with predicates that determine whether a token from *i*-th input place can go to the *j*-th output place; an index matrix with natural numbers that determine the capacities of the arc between *i*-th input and *j*-th output place; and a special condition, that determines whether the transition can be activated. The GN definition is also more complex than the definition of a PN. The GN-tokens enter the net with initial characteristics, determined by the characteristic function *X*. Upon entering a new place, the GN-tokens obtain new characteristics, defined by the characteristic function Φ. In contrast to the Colored PNs and the Predicate-Transition Nets, the GN-tokens can keep all their characteristics and they can be used for evaluation of the truth-values of the transition condition predicates [1, 2].

Definition 3 (*GN loop*). A sequence of places of a GN, which a given token can go through sequentially and can reach the starting position, is called a *GN loop*.

III. Verification of GNs, based on Floyd's Method of Verification of a Flowchart

The method presented here can be used for verification of GN models without temporal component (without the components: θ_1, θ_2, θ_K, T, t^o and t^*). Respectively, the GN transitions do not contain the temporal components t_1 and t_2. The reason for this is that this restriction and the existence of index matrices with predicates and characteristic functions cause each component of a GN to have a respective segment as in a flowchart. The GNs, which will be verified through a technique following Floyd's method for verification of flowcharts [12], consist of the components given in Fig. 2. In order to increase the readability, we will use the notations for Floyd's method for verification of flowcharts.

Three types of characteristics of the GN (grouped as three vectors) are distinguished:

- An *input* vector $\bar{x} = (x_1, x_2, ..., x_n)$, which sets the values of the initial characteristics $x_1, x_2, ..., x_n$ of the GN tokens.
- An *intermediate* vector $\bar{y} = (y_1, y_2, ..., y_m)$, which is used as temporary storage during the GN execution and describes the changing values of the characteristics $y_1, y_2, ..., y_m$ of the GN tokens.
- An *output* vector $\bar{z} = (z_1, z_2, ..., z_k)$, that yields the values of output characteristics $z_1, z_2, ..., z_k$ of tokens when the GN execution terminates.

These characteristics are specific for each GN model under verification.

Three types of non-empty domains are distinguished as well:

- an input domain $D_{\bar{x}} = D_{x1} \times D_{x2} \times \ldots \times D_{xn}$;
- an intermediate domain $D_{\bar{y}} = D_{y1} \times D_{y2} \times \ldots \times D_{ym}$ and
- an output domain $D_{\bar{z}} = D_{z1} \times D_{z2} \times \ldots \times D_{zk}$

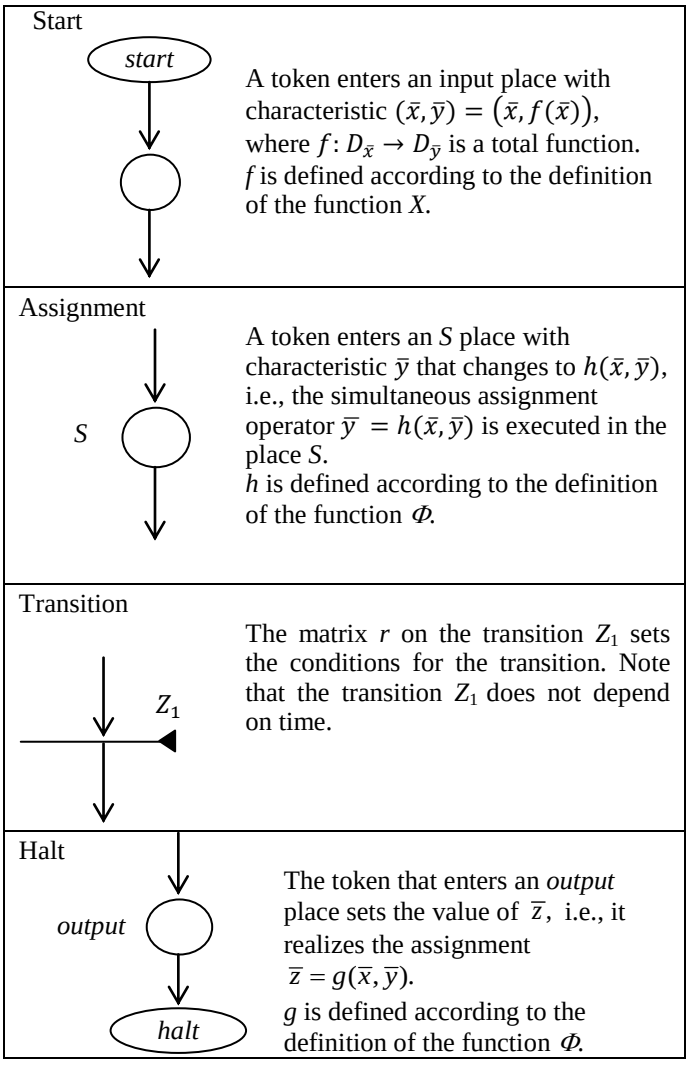

Fig. 2. Components of a GN.

As in the case of the flowcharts, the verification of a GN without temporal components depends on the following predicates:

- An *input* predicate, which will be denoted by $\varphi(\bar{x})$. It is a total predicate over $D_{\bar{x}}$, which describes those elements (data) that may be used as values of the initial characteristics $x_1, x_2, ..., x_n$ of the GN tokens.
- An *output* predicate, which will be denoted by $\psi(\bar{x}, \bar{z})$. It is a total predicate over $D_{\bar{x}} \times D_{\bar{z}}$, which describes the relationships that must be satisfied between the input and the output values of the characteristics of the GN tokens at the termination of the GN execution.

The predicates $\varphi(\bar{x})$ and $\psi(\bar{x}, \bar{z})$ set the input-output specification with respect to which the GN will be verified.

Definition 4. GN *P* without temporal components is *partially correct with respect to* $\varphi(\bar{x})$ *and* $\psi(\bar{x}, \bar{z})$ if for every input vector $\bar{\varepsilon}$, for which the input predicate $\varphi(\bar{\varepsilon})$ is *true* and the computation of *P* terminates, $\psi(\bar{\varepsilon}, P(\bar{\varepsilon}))$ is *true*.

Definition 5. GN *P* without temporal components *terminates over* $\varphi(\bar{x})$, if for every input vector $\bar{\varepsilon}$, for which $\varphi(\bar{\varepsilon})$ is *true*, the execution of *P* terminates (stops).

Definition 6. GN *P* of the type described above is *totally correct with respect to* $\varphi(\bar{x})$ *and* $\psi(\bar{x}, \bar{z})$, if for every vector $\bar{\varepsilon}$, for which the input predicate $\varphi(\bar{\varepsilon})$ is *true*, the execution of *P* terminates over $\varphi(\bar{x})$ and the output predicate $\psi(\bar{\varepsilon}, P(\bar{\varepsilon}))$ is *true*.

Verifying a GN without temporal components with respect to the input-output specification means proving its total correctness regarding this specification. What follows from the definitions above is that the GN *P* is totally correct with respect to $\varphi(\bar{x})$ and $\psi(\bar{x}, \bar{z})$, if *P* is partially correct with respect to $\varphi(\bar{x})$ and $\psi(\bar{x}, \bar{z})$ and *P* terminates over $\varphi(\bar{x})$.

A. Partial Correctness of a GN

A technique for proving that a GN of the type described above is partially correct with respect to input predicate $\varphi(\bar{x})$ and output predicate $\psi(\bar{x}, \bar{z})$ will be presented. It is similar to this for flowcharts [12].

Let us execute the following three steps:

Step 1 (Cutpoints).

Each GN loop connects to a cut (see cutpoints S_1 and S_2 on Fig. 5). *Start* and *halt* cuts are added to this set of cutpoints (see Fig. 3). Only paths which start and end at cutpoints and which have no intermediate cutpoints are considered. For each path α from cutpoint *i* to cutpoint *j* there is a predicate $R_\alpha(\bar{x}, \bar{y})$ over $D_{\bar{x}} \times D_{\bar{y}}$ and a vector $r_\alpha(\bar{x}, \bar{y})$

$$r_\alpha : D_{\bar{x}} \times D_{\bar{y}} \to D_{\bar{y}}.$$

The predicate $R_\alpha(\bar{x}, \bar{y})$ indicates the condition for this path to be traversed, and the vector $r_\alpha(\bar{x}, \bar{y})$ describes the transformation of the values of $\bar{y}$ affected by path α traversion.

This function can be derived by means of the *backward-substitution technique* [12]. First, the values of the $R_\alpha(\bar{x}, \bar{y})$ and $r_\alpha(\bar{x}, \bar{y})$ (values in cut *j*), are set to *true* (we will denote *true* by *T*) and $\bar{y}$, respectively. Then, at each step, the old *R* and *r* are used to construct the new *R* and *r*, moving backwards toward the cutpoint *i*. The description of the new values in the components is shown in Fig. 3. The resulting *R* and *r* in the cut *i* are the desired $R_\alpha(\bar{x}, \bar{y})$ and $r_\alpha(\bar{x}, \bar{y})$.

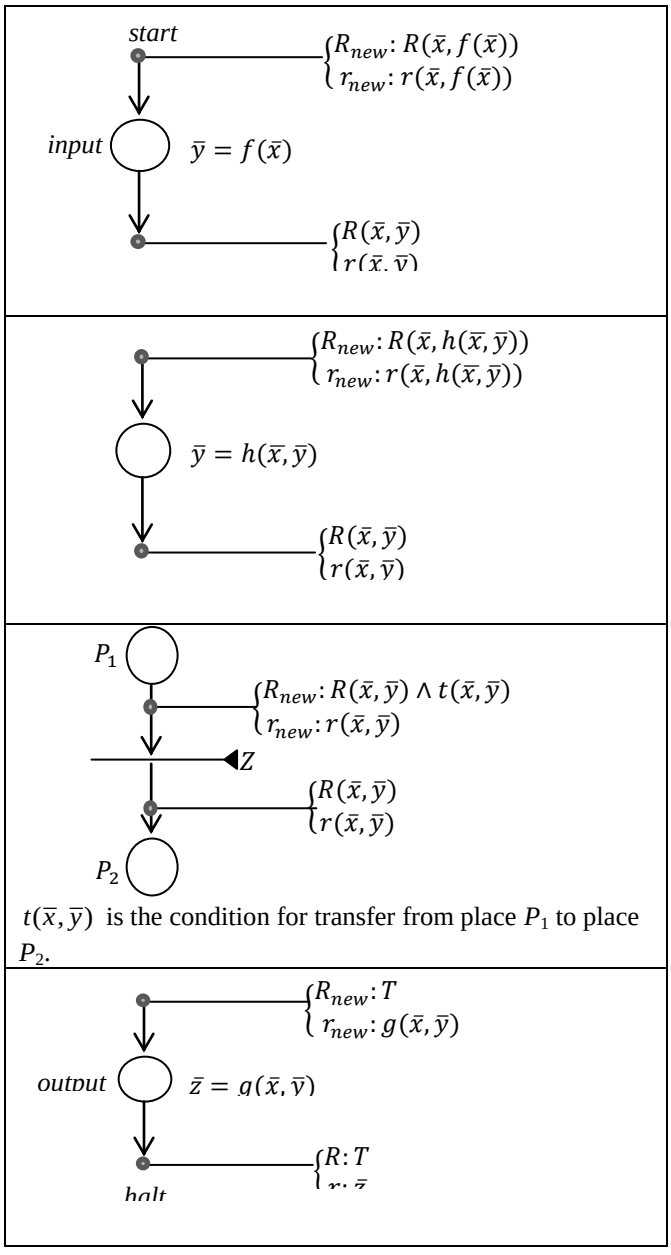

Fig. 3. Rules for constructing the new predicate $R(\bar{x}, \bar{y})$ and the new function $r(\bar{x}, \bar{y})$ for different types of components.

Fig. 4 presents an example for constructing $R_\alpha(\bar{x},\bar{y})$ and $r_\alpha(\bar{x},\bar{y})$ for path α, where the condition for transfer from the input place P_1 to the output place P_2 is $t_1(\bar{x},\bar{y})$, and the condition for transfer from the input place P_2 to the output place P_3 is $t_2(\bar{x},\bar{y})$.

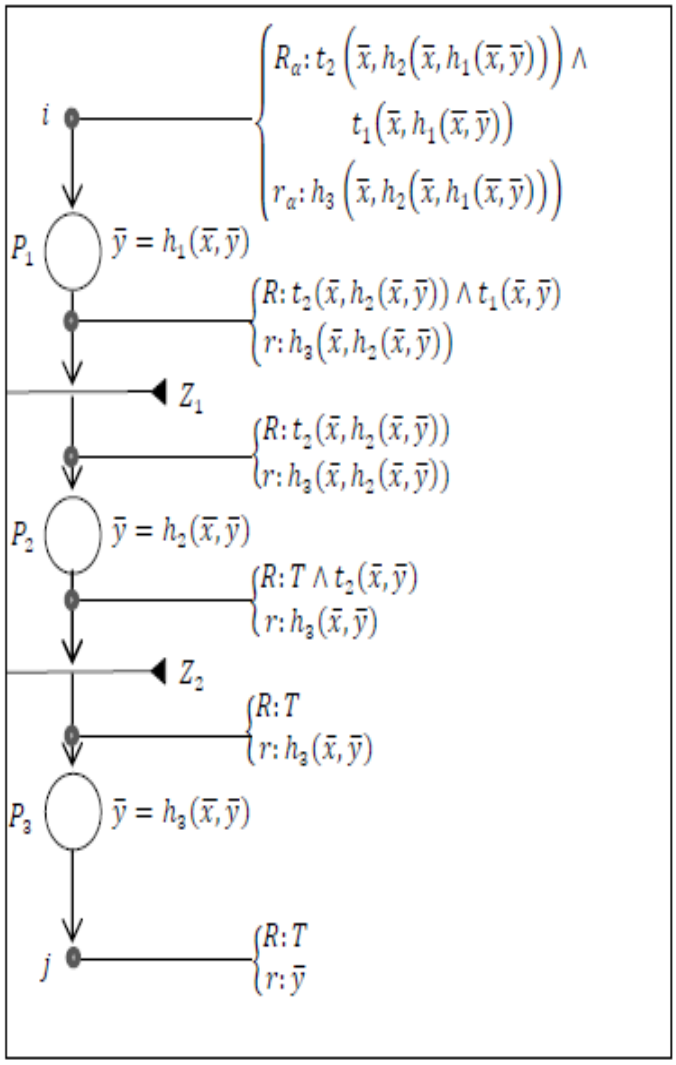

Fig. 4. Constructing the R_α and r_α functions for the path α.

Step 2 (Inductive assertions).

With each cutpoint *i* of the GN, a predicate $p_i(\bar{x},\bar{y})$ is associated. This predicate is called *inductive assertion*. It characterizes the relation between the values of the characteristics $\bar{x}$ and $\bar{y}$ of the tokens at this point, i.e., $p_i(\bar{x},\bar{y})$ will have the property that, whenever the implementation reaches point *i*, $p_i(\bar{x},\bar{y})$, must be *true* for the current values of $\bar{x}$ and $\bar{y}$ at this point. The input predicate $\varphi(\bar{x})$ is attached to the *start* cutpoints, and the output predicate $\psi(\bar{x},\bar{z})$ is attached to the *halt* cutpoints.

Step 3 (Verification conditions).

The final step is to build the verification conditions for each path of the GN:

- For each path α for which *i* is cutpoint *start*

$$\forall\bar{x}\left[\varphi(\bar{x}) \wedge R_\alpha(\bar{x}) \Rightarrow p_j\left(\bar{x}, r_\alpha(\bar{x})\right)\right] \quad (1)$$

- For each path α, from the *i* cutpoint to cutpoint *j*

$$\forall\bar{x}\forall\bar{y}\,[p_i(\bar{x},\bar{y}) \wedge R_\alpha(\bar{x},\bar{y}) \Rightarrow p_j(\bar{x}, r_\alpha(\bar{x},\bar{y}))] \quad (2)$$

- For each path α for which *j* is cutpoint *halt*

$$\forall\bar{x}\forall\bar{y}\,[p_i(\bar{x},\bar{y}) \wedge R_\alpha(\bar{x},\bar{y}) \Rightarrow \psi(\bar{x}, r_\alpha(\bar{x},\bar{y}))] \quad (3)$$

and to prove that all these conditions are *true*.

If the constructed verification conditions for all paths that cover the GN are satisfied, the GN is partially correct with respect to $\varphi(\bar{x})$ and $\psi(\bar{x},\bar{z})$. This leads to the follow theorem.

Theorem 1. The following steps are applied to a given GN *P* without temporal components, an input predicate $\varphi(\bar{x})$ and an output predicate $\psi(\bar{x},\bar{z})$:

- The loops of the GN are cut.
- An appropriate set of inductive assertions is found.
- The verification conditions (1), (2) and (3) are constructed.

If all the verification conditions are true, then *P* is partially correct with respect to $\varphi(\bar{x})$ and $\psi(\bar{x},\bar{z})$.

B. Termination of a GN

The following is a description of a method of proving the termination of a GN without temporal components regarding an input predicate $\varphi(\bar{x})$. The method was proposed by Floyd for a flowchart [12]. Well-founded sets are used [12].

Note that the paths in steps 1 and 3 do not contain intermediate cutpoints.

Let's perform the following three steps:

Step 1 (Good assertions).

Select a set of cutpoints that cut the loops of the GN. Associate an assertion $q_i(\bar{x},\bar{y})$ with every cutpoint *i*, which is a good assertion [12], i.e.,

- For each path α from the *start* cutpoint to cutpoint *j*, the following is satisfied:

$$\forall\bar{x}\left[\varphi(\bar{x}) \wedge R_\alpha(\bar{x}) \Rightarrow q_j\left(\bar{x}, r_\alpha(\bar{x})\right)\right] \quad (4)$$

- For each path α from the *i* cutpoint to cupoint *j*, the following is satisfied:

$$\forall\bar{x}\forall\bar{y}\left[q_i(\bar{x},\bar{y}) \wedge R_\alpha(\bar{x},\bar{y}) \Rightarrow q_j\left(\bar{x}, r_\alpha(\bar{x},\bar{y})\right)\right] \quad (5)$$

Step 2 (Well-founded set).

Choose a well-founded set $(W, \prec)$ and with every cutpoint i of the GN associate a partial function $u_i(\bar{x},\bar{y})$

$$u_i:\ D_{\bar{x}} \times D_{\bar{y}} \longrightarrow W$$

which is a good function [12], i.e., for every curpoint i, is satisfied:

$$\forall\bar{x}\forall\bar{y}[q_i(\bar{x},\bar{y}) \Rightarrow (u_i(\bar{x},\bar{y}) \in W)] \quad (6)$$

Step 3 (Termination conditions).

Show that the termination conditions hold. This means that for every path α from a cutpoint i to a cutpoint j, which is a part of some GN loop, the following is satisfied:

$$\forall\bar{x}\forall\bar{y}\left[q_i(\bar{x},\bar{y}) \wedge R_\alpha(\bar{x},\bar{y}) \Rightarrow \left(u_i(\bar{x},\bar{y}) \succ u_j\left(\bar{x}, r_\alpha(\bar{x},\bar{y})\right)\right)\right] \quad (7)$$

This means that after each time a path, which is a part of a loop, is executed, the values of the functions u_i, that are associated with the cuts, strictly decrease. As $(W, \prec)$ is a well-founded set, i.e. there are no infinite decreasing sequences of elements of W, then the number of the path executions is limited. This leads to the follow theorem.

Theorem 2. The following steps are applied to a given GN P of the type described above and an input predicate $\varphi(\bar{x})$:

- The loops are cut and "good" (satisfying (4) and (5)) inductive assertions are found.
- A well-founded set is selected and "good" (satisfying (6)) partial functions are found.
- The termination conditions (7) are checked.

If all the termination condition are true, then P terminates over φ.

IV. Illustrative Example

The generalized net in Fig. 5 implements the model of a sequential program, finding $z = x_1^{x_2}$ (where 0^0 is considered to be equal to 1), where x_1 is an integer and x_2 is a nonnegative integer.

A start component, a halt component, 2 assignment components and 9 transition components are shown in the figure. The execution of the GN begins at entering the token with characteristic

$$(\bar{x},\bar{y}) = (x_1, x_2, y_1, y_2, y_3),$$

into an *input* place, where y_1 means the current value of the base, y_2 means the current value of the exponent and y_3 – the current value of the exponential result.

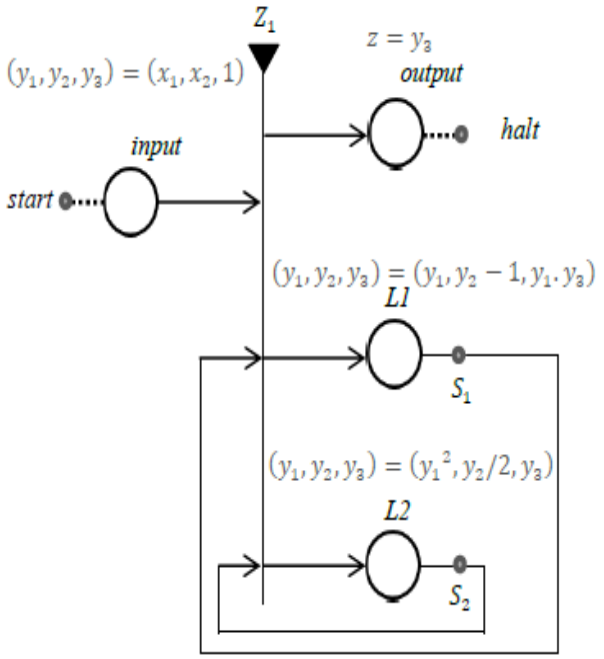

Fig. 5. GN which implements finding $z = x_1^{x_2}$.

The moment the token enters an *input* place with characteristic $(\bar{x},\bar{y}) = (x_1, x_2, y_1, y_2, y_3)$, it gets the value $(x_1, x_2, x_1, x_2, 1)$. The figure also shows how $\bar{y} = (y_1, y_2, y_3)$ of the characteristic of the token changes in places L_1 and L_2; and shows that, when it enters into an *output* place, it receives characteristic $z = y_3$.

The GN has one transition Z_1, with predicate matrix r_1 of transition:

r_1	*output*	$L1$	$L2$
input	$y_2 = 0$	$y_2 \neq 0 \wedge odd(y_2)$	$y_2 \neq 0 \wedge \neg odd(y_2)$
$L1$	$y_2 = 0$	$y_2 \neq 0 \wedge odd(y_2)$	$y_2 \neq 0 \wedge \neg odd(y_2)$
$L2$	$y_2 = 0$	$y_2 \neq 0 \wedge odd(y_2)$	$y_2 \neq 0 \wedge \neg odd(y_2)$

We will perform GN verification by the method described above over:

- the input predicate: $\varphi(\bar{x}) : x_2 \geq 0$ and
- the output predicate: $\psi(\bar{x},\bar{z}) : z = x_1^{x_2}$

where $\bar{x} = (x_1, x_2), \bar{y} = (y_1, y_2, y_3), \bar{z} = z$.

A. Partial Correctness

Let us cut the two loops of the GN at points S_1 and S_2 (see Fig. 5), and attach to the cutpoint S_1 and cutpoint S_2 the assertion:

$$p(\bar{x},\bar{y}): x_2 \geq 0 \wedge y_2 \geq 0 \wedge y_3.y_1^{y_2} = x_1^{x_2}.$$

This GN is covered by the following paths:

1. start → halt	4. $S_1 \rightarrow S_1$	7. $S_2 \rightarrow S_1$
2. start → S_1	5. $S_2 \rightarrow S_2$	8. $S_1 \rightarrow$ halt
3. start → S_2	6. $S_1 \rightarrow S_2$	9. $S_2 \rightarrow$ halt

We will prove the verification conditions only for paths 1 and 7 below. The other verification conditions are proved likewise. The functions *R* and *r* for the paths 1, 2, …, 9 have the form:

1. *R*: $x_2 = 0$, *r*: 1
2. *R*: $x_2 \neq 0 \wedge odd(x_2)$, *r*: (x_1, x_2-1, x_1)
3. *R*: $x_2 \neq 0 \wedge \neg\, odd(x_2)$, *r*: $(x_1^2, \frac{x_2}{2}, 1)$
4. *R*: $y_2 \neq 0 \wedge odd(y_2)$, *r*: $(y_1, y_2-1, y_1.y_3)$
5. *R*: $y_2 \neq 0 \wedge \neg\, odd(y_2)$, *r*: $(y_1^2, \frac{y_2}{2}, y_3)$
6. *R*: $y_2 \neq 0 \wedge \neg\, odd(y_2)$, *r*: $(y_1^2, \frac{y_2}{2}, y_3)$
7. *R*: $y_2 \neq 0 \wedge odd(y_2)$, *r*: $(y_1, y_2-1, y_1.y_3)$
8. *R*: $y_2 = 0$, *r*: y_3
9. *R*: $y_2 = 0$, *r*: y_3

The verification condition for path 1 has the form:

$$\varphi(\bar{x}) \wedge (x_2 = 0) \Rightarrow \psi(\bar{x}, 1),$$

i.e.,

$$(x_2 \geq 0) \wedge (x_2 = 0) \Rightarrow 1 = x_1^{x_2}$$

is evidently satisfied.

The verification condition for path 7 has the form:

$$p(\bar{x},\bar{y}) \wedge y_2 \neq 0 \wedge odd(y_2) \Rightarrow p(\bar{x}, y_1, y_2 - 1, y_1.y_3)$$

i.e.,

$$x_2 \geq 0 \wedge y_2 \geq 0 \wedge y_3.y_1^{y_2} = x_1^{x_2} \wedge y_2 \neq 0 \wedge odd(y_2) \Rightarrow$$

$$x_2 \geq 0 \wedge (y_2 - 1) \geq 0 \wedge y_1 y_3.y_1^{y_2 - 1} = x_1^{x_2}$$

Since all verification conditions are true, it follows that the GN is partially correct.

B. Termination

Let choose the well-founded set (*N*, <), that is, the set of natural numbers with the usual ordering <. We cut the two loops at points S_1 and S_2 (see Fig. 5.) Next, we choose $q_{s_1}(\bar{x},\bar{y})$ and $q_{s_2}(\bar{x},\bar{y})$ to be $y_2 \geq 0$. Let $u_{s_1}(\bar{x},\bar{y})$ and $u_{s_2}(\bar{x},\bar{y})$ are equal to y_2.

Step 1. $q_{s_1}(\bar{x},\bar{y})$ and $q_{s_2}(\bar{x},\bar{y})$ are good assertions.

Here we will prove the condition only for path 6. The assertions for other paths are proved similarly.

$$q_{S_1}(\bar{x}, y_1, y_2, y_3) \wedge y_2 \neq 0 \wedge \neg odd(y_2) \Rightarrow q_{S_2}(\bar{x}, y_1^2, \frac{y_2}{2}, y_3)$$

i.e.,

$$y_2 \geq 0 \wedge y_2 \neq 0 \wedge \neg odd(y_2) \Rightarrow \frac{y_2}{2} \geq 0$$

Step 2. $u_{s_1}(\bar{x},\bar{y})$ and $u_{s_2}(\bar{x},\bar{y})$ are good functions.

The condition for cutpoint S_1 is:

$$q_{S_1}(\bar{x},\bar{y}) \Rightarrow \left(u_{S_1}(\bar{x},\bar{y}) \in N \right)$$

that is, $y_2 \geq 0 \Rightarrow y_2 \geq 0$. The condition is evidently satisfied.

The condition for S_2 cutpoint is proved similarly.

Step 3. The termination condition holds.

Only paths 4, 5, 6 and 7 are considered as they are parts of some GN loop.

Path 4 $S_1 \rightarrow S_1$

$$q_{S_1}(\bar{x}, y_1, y_2, y_3) \wedge y_2 \neq 0 \wedge odd(y_2) \Rightarrow$$

$$(u_{S_1}(\bar{x}, y_1, y_2, y_3) > u_{S_1}(\bar{x}, y_1, y_2 - 1, y_1.y_3))$$

i.e.,

$$y_2 \geq 0 \wedge y_2 \neq 0 \wedge odd(y_2) \Rightarrow y_2 > y_2 - 1$$

Path 5 $S_2 \rightarrow S_2$

$$q_{S_2}(\bar{x}, y_1, y_2, y_3) \wedge y_2 \neq 0 \wedge \neg odd(y_2) \Rightarrow$$

$$(u_{S_2}(\bar{x}, y_1, y_2, y_3) > u_{S_2}(\bar{x}, y_1^2, \frac{y_2}{2}, y_3))$$

i.e.,

$$y_2 \geq 0 \wedge y_2 \neq 0 \wedge \neg odd(y_2) \Rightarrow y_2 > \frac{y_2}{2}$$

Path 6 $S_1 \rightarrow S_2$

$$q_{S_1}(\bar{x}, y_1, y_2, y_3) \wedge y_2 \neq 0 \wedge \neg odd(y_2) \Rightarrow$$

$$(u_{S_1}(\bar{x}, y_1, y_2, y_3) > u_{S_2}(\bar{x}, y_1^2, \frac{y_2}{2}, y_3))$$

i.e.,

$$y_2 \geq 0 \wedge y_2 \neq 0 \wedge \neg odd(y_2) \Rightarrow y_2 > \frac{y_2}{2}$$

Path 7 $S_2 \rightarrow S_1$

$$q_{S_2}(\bar{x}, y_1, y_2, y_3) \wedge y_2 \neq 0 \wedge odd(y_2) \Rightarrow$$

$$(u_{S_2}(\bar{x}, y_1, y_2, y_3) > u_{S_1}(\bar{x}, y_1, y_2 - 1, y_1.y_3))$$

i.e.,

$$y_2 \geq 0 \wedge y_2 \neq 0 \wedge odd(y_2) \Rightarrow y_2 > y_2 - 1$$

Since all conditions of the three steps are *true*, the GN terminates for every $x_2 \geq 0$ natural number.

V. Applications and Ideas for Further Research

The necessity for developing methods and environments for formal verification of GN models is triggered by the implementation of real-life methods of software protection by using GNs [13]. Using Floyd's inductive assertion method for verification of flowcharts, adapted for GNs, GN models which realize sequential processes, and sequential programs in particular [14], can be verified. It can also be applied for verification of GNs, which model parallel processes. To this end, a transition's type component □ is applied.

Following our belief that training in applying formal methods for developing of correct software is the most efficient method of implementing these methods in the software industry, we intend on introducing the method described above in teaching students of specialty Informatics and Computer Sciences at Sofia University. In order to achieve this, we will develop and add tools for its application to the educational framework presented in [15]. The ideas presented in [16, 17, 18] are implemented in the educational framework. The resulting educational framework may be applied not only in teaching programming and data structures [19, 20], but also in creating GN models of applications, such as: classical transaction processing systems [21], mobile information applications for public access [22, 23], business process models [24, 25], software services models [26, 27], data models [28], etc. Thus, students will be stimulated to search for out of the box solutions of the tasks given in the programming courses [29] and in the courses in discrete mathematics (discrete structures) [30].

Furthermore, we consider designing and improving an educational framework with tools for verification of GN models with temporal components.

VI. Related Work

The field of formal verification of GN models has not been studied by now. The most closely related work with the one presented here is the book by Zohar Manna [12]. This book represents an introduction in the mathematical theory of informatics, and is considered the main reference book in this area for many universities around the world, including the Sofia University. Chapter 3 of [12] explores the verification of computer programs. Sections 3-1 and 3-2 of this chapter contains definitions, related to the verification of programs presented in terms of flowcharts, and Floyd's inductive assertions method for program verification is provided. Since the methodology of development of Generalized Net models is difficult enough as such, in order to enhance the understanding of the herewith presented adaptation of Floyd's inductive assertions method for formal verification of GN models without temporal components, we opted to use as much as possible the denotations and theorems, formulated in [12].

VII. Conclusion

The paper presents the authors' first attempt to achieve formal verification of GN models. Our research has been restricted to verification of GN models without temporal components. Since the proposed verification approach belongs to the set of formal verification methods, it bears all of their limitations, as well. Significant efforts are required to construct the input/output specification; to prove the conditions for partial correctness and for termination of the execution; the procedures are to be realized by the rare highly qualified experts in formal modelling. Using automated tools for formal theorem proving would reduce the mentioned difficulties. The active and impactful research on development of such tools has motivated us to continue the work in this direction by developing methods for formal verification of GN models featuring temporal components, as well as verification of the GN models of the applications proposed here in Part V.

References

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

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

[3] T. Trifonov, 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] T. Trifonov, K. Georgiev, and K. Atanassov, "Software for modelling with Generalised Nets," Issues in Intuitionistic Fuzzy Sets and Generalized Nets, vol. 6, pp. 36–42, 2008.

[5] V. Gochev, "An implementation of generalized nets using object-oriented programming in .NET framework, Management and education," vol. VI (4), pp. 227–231, 2010.

[6] K. Atanassov, D. Dimitrov, and V. Atanassova, "Algorithms for Tokens Transfer in the Different Types of Intuitionistic Fuzzy Generalized Nets," Cybernetics and Information Technologies, vol. 10, No. 4, pp. 22–35, 2010.

[7] D. G. Dimitrov, "Graphical Environment for Modeling and Simulation with Generalized Nets," Annual of "Informatics", Section Union of Scientists in Bulgaria, vol. 3, pp. 51–66, 2010.

[8] D. G. Dimitrov, "Software Products Implementing Generalized Nets," Annual of "Informatics", Section Union of Scientists in Bulgaria, vol. 3, pp. 37–50, 2010.

[9] D. G. Dimitrov, "Optimized Algorithm for Token Transfer in Generalized Nets," Recent Advances in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics, vol. 1, pp. 63–68, 2010.

[10] N. Angelova, M. Todorova, and K. Atanassov, "GN IDE: Implementation, Improvements and Algorithms," Comptes Rendus de L'Academie Bulgare des Sciences, Tome 69, vol. 4, pp. 411−420, 2016.

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

[12] Z. Manna, Mathematical theory of computation, New York, 1974.

[13] M. Todorova, and D. Orozova, "Software protection integrating registration-number and anti-debugging protections," 9th International Conference Information Systems & Grid Technologies, St. Kliment Ohridski University Press, 2015.

[14] M. Todorova, and D. Orozova, "Generalized Net Model of Sequential Programs," 20th International Symposium on Electrical Apparatus and Technologies (SIELA), 3 – 6 June, 2018, Bourgas, Bulgaria (in print), [Digests 20th International Symposium on Electrical Apparatus and Technologies, Bulgaria, pp. 265−266, 2018].

[15] M. Todorova and K. Kanev, "Educational framework for verification of object−oriented programs," The Joint International Conference on Human−Centered Computer Environments HCCE'2012, Hamamatsu, Japan, pp. 23−27, 2012.

[16] A. Shannon, D. Langova−Orozova, E. Sotirova, I. Petrounias, K. Atanassov, M. Krawczak, P. Melo−Pinto, and T. Kim, Generalized Net Modelling of University Processes, KvB Visual Concepts Pty Ltd, Monograph No. 7, Sydney, 2005.

[17] A. Shannon, K. Atanassov, D. Orozova, M. Krawczak, E. Sotirova, P. Melo−Pinto, I. Petrounias and T. Kim, Generalized nets and

information flow within a university, Warsaw School of Information Technology, Warsaw, 2007.

[18] D. Orozova, and K. Atanassov, "Generalized net model of the process of selection and usage of an intelligent e-learning system," Comptes Rendus de l'Academie bulgare des Sciences, tome 65, No. 5, pp. 591−598, 2012.

[19] I. Donchev, and E. Todorova, "Implementation of ADS Linked List Via Smart Pointers," International Journal of Advanced Computer Science and Applications, vol. 6, No. 2, pp. 196−203, 2015.

[20] I. Donchev, and E. Todorova, "Implementation of Binary Search Trees Via Smart Pointers," International Journal of Advanced Computer Science and Applications, vol. 6, No. 3, pp. 59−64, 2015.

[21] K. Kaloyanova, "Successful practices for learning information systems development," 7th International Technology, Education and Development Conference, pp. 4849−4855, 2013.

[22] I. Patias and V. Georgiev, "Mobile medical applications as instrument in supporting patients compliance," American Journal of Engineering Research, vol. 6, No. 8, pp. 96−102, 2017.

[23] I. Patias and V. Georgiev, "Modeling and implementation of bus rapid transit corridor based on isolated or coordinated traffic prioritization and automatic location," Journal of Emerging Research and Solutions in ICT, vol. 1, No. 2, pp. 17–24, 2016.

[24] E. Krastev and K. Shahinyan, "Computer assisted quality assessment of a set of business process models," Proceedings of the 9th IEEE European Modelling Symposium of Mathematical Modelling and Computer Simulation, IEEE Computer Society, pp. 180−186, 2015.

[25] V. Dimitrov, "Deriving semantics from WS-BPEL specifications of parallel business processes on an example," Computer Research and Modeling, vol. 7, No. 3, pp. 445−454, 2015.

[26] S. Hadzhikoleva and E. Hadzhikolev, "Model for automated integration of data from heterogeneous sources in the COMPASS-OK application for (self) evaluation and accreditation," International Journal of Applied Engineering Research, vol. 11, No. 12, pp 7648−7653, 2016.

[27] S. Hadzhikoleva, T. Rachovski and E. Hadzhikolev, "Generalized Net Model for Building Responsive Design of Web Pages," 20th International Symposium on Electrical Apparatus and Technologies SIELA 2018, 3 − 6 June 2018, Bourgas, Bulgaria (in print).

[28] V. Dimitrov, "Relationship Specified in Z-notation," Physics of Elementary Particles and Atomic Nuclei, Letters, vol. 8, No. 4 (167), pp. 655−663, 2011.

[29] K. Yordzhev, "The Bitwise Operations Related to a Fast Sorting Algorithm," International Journal of Advanced Computer Science and Applications, vol. 4, No. 9, pp. 103−107, 2013.

[30] K. Yordzhev, Sudoku, S-permutation matrices and bipartite graphs, LAP LAMBERT Academic Publishing, 2016.