12th Int. Workshop on Generalized Nets
Burgas, 17 June 2012, 73–78

# Correctness of a formal generalized net project of a class of an object-oriented program

**Magdalina Todorova**

Faculty of Mathematics and Informatics, Sofia University
5 "J. Bouchier" Str., Sofia, Bulgaria
e-mail: todorova_magda@hotmail.com

**Abstract:** The article defines formal generalized net project of a class, correctness of a formal generalized net project of a class in respect to a defined for it specification and correspondence of the formal generalized net project to the class realization. A theorem is formulated to define criteria for correctness of a formal generalized net project of a class in respect to the defined for it specification.
**Keywords:** Generalized nets, Object-oriented programming, Program correctness, Modeling, Verification.
**AMS Classification:** 68Q85, 68N19.

## 1 Introduction

An approach to verification of object-oriented programs (OOP) is described in [6]. It is based on building generalized net (GN)-models [1, 2] of the programs (functions) and of the specifications, according to which verification and checking of correspondence (consistency) of the two models are performed. The algorithm for checking of correspondence is also defined as a generalized net, which executes the models of the program and of the specification in parallel. A choice has been made for the OOP to contain only one class, in order to simplify the presentation.

An important component of the described approach of OOP verification is the model 'design by contract', [4, 5]. It includes description of specific statements (contracts), which must hold in specific places of the programs. The contracts may describe: basic statements, class invariants, preconditions and postconditions of general functions and member functions of classes. They are also called *specification* of the program.

Class specification of an object-oriented program is given by defining the invariant of the class, the preconditions and the postconditions of the member functions of the class. If the member functions of the class use operators for a cycle, an invariant and a loop termination function are also to be defined for each one of them. Each object of a class satisfies properties that are defined by a predicate called class invariant. The class invariant must hold after the execution of each constructor, as well as before and after the execution of each method of the class.

Informally, a class is correct in respect to a defined for it specification when its realization corresponds to the preconditions, postconditions and the class invariant. The correctness is formulated more precisely in [4] as follows. Let *C* is a class, *Inv* is its invariant, and *r* is an arbitrary member function of the class. $Body_r$ denotes the body of *r*, $pre_r(x_r)$ and $post_r(x_r)$ are the precondition and postcondition of *r*, with admissible arguments $x_r$. $Default_C$ denotes a statement which expresses implicit relations among data members of the class *C*.

*Definition 1*: Class *C* is correct in respect to its specification [4] if:

a) The Hoare's triple

$$\{\ Default_C \wedge pre_p(x_p)\ \}\ Body_P\ \{\ post_p(x_p) \wedge Inv\ \} \tag{1}$$

holds for each class constructor *P* and for each admissible set of arguments $x_p$.

b) The Hoare's triple

$$\{\ pre_r(x_r) \wedge Inv\ \}\ Body_r\ \{\ post_r(x_r) \wedge Inv\ \} \tag{2}$$

holds for each member function *r*, different from the class constructor, with a set of admissible arguments $x_r$.

The specification, in respect to which the described in [6] OOP verification is performed, defines some or all admissible sequences of calls to the member functions of the OOP class. It is given via a generalized net and is also called a formal GN-project of the class. An important moment after building the GN-model of the class, which defines the specification, is the verification of its correctness. The latter is the research subject of this article.

## 2 Correctness of a formal GN-project of a class

Let a generalized net consists of one transition as presented in Figure 1 (a) and a set of transitions as in Figure 1 (b).

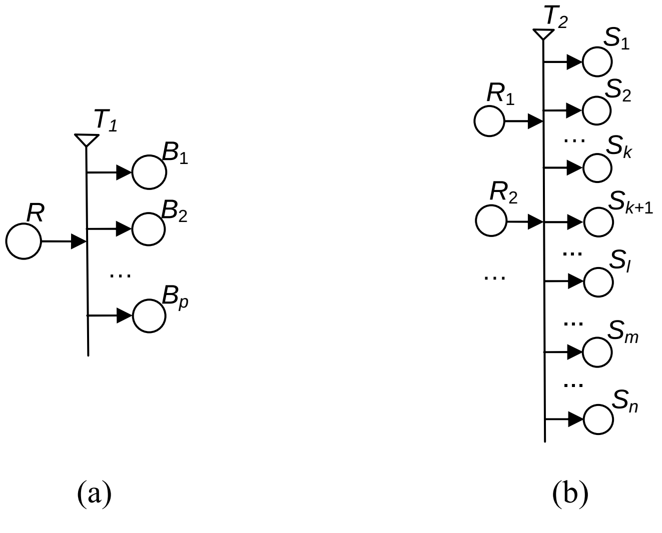

Figure 1: Definition of transitions of a generalized net, presenting a formal GN-project of a class

Here,

$$T_1 = \langle \{R\}, \{B_1, B_2, \ldots, B_p\}, t_1' \rangle$$

$$T_2 = \langle \{R_1, R_2, \ldots\}, \{S_1, S_2, \ldots, S_n\}, t_2' \rangle$$

$$t_1' = \begin{array}{c|cccc} & B_1 & B_2 & \ldots & B_p \\ \hline R & V_1 & V_2 & \ldots & V_p \end{array},$$

$$t_2' = \begin{array}{c|ccccccccccc} & S_1 & S_2 & \ldots & S_k & S_{k+1} & \ldots & S_l & \ldots & S_m & \ldots & S_n \\ \hline R_1 & W_1 & W_2 & \ldots & W_k & W_{k+1} & \ldots & W_l & \ldots & W_m & \ldots & W_n \\ R_2 & W_1 & W_2 & \ldots & W_k & W_{k+1} & \ldots & W_l & \ldots & W_m & \ldots & W_n \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{array},$$

where

- $V_s$ = *"Member-function is the constructor $C_s$"* $\wedge$ *"Condition $Q_s$ holds* $(1 \le s \le p)$*"*
- $W_i$ = *"Member-function is f"* $\wedge$ *"Condition $P_i$ holds* $(1 \le i \le k)$*"*
- $W_j$ = *"Member-function is g"* $\wedge$ *"Condition $P_j$ holds* $(k + 1 \le j \le l)$*"*
- …
- $W_t$ = *"Member-function is h"* $\wedge$ *"Condition $P_t$ holds* $(m \le t \le n)$*"*

The predicates $P_1, P_2, \ldots, P_k$, as well as $P_{k+1}, P_{k+2}, \ldots, P_l$ and … $P_m, P_{m+1}, \ldots, P_n$, which refer to the application of one and the same member function of the class, are mutually exclusive (only one of them can have a *true* value at a time). In addition, *f*, *g*, …, *h* are different member functions of the class.

Each transition place of the formal GN project of a class presents a logical state in which a class object can take. It is called *a logical state of an object in a place*, or *a logical state of a place*. The logical state is given via a Boolean expression. The tokens in the $GN_C$ places correspond to objects of the class. *A set of data*, defined by member data of the class, refers to each class object. The token characteristic is given by a pair of the type (*object, place*). Here *object* is an identification giving a name to the class object; and *place* is the name of the place occupied by the token (object) in $GN_C$. In addition to these two components, the data set of the object is implicitly included in the token characteristic through the parameter *object* (called also *token data*); the logical state of the object in the place is respectively included through the parameter *position.*

*Definition 2*: The generalized net, as defined above, is a formal GN-project of a class, if the following conditions hold for it:

i. The implication

$$\textit{logical state of the place } S \Rightarrow \textit{pre}_q \tag{3}$$

holds for each place *S* of $GN_C$, different from the input place. Here *q* is an arbitrary member function of a class, which is part of the conditions of the transition for which *S* is an input place.

The implication

$$\textit{Default}_C \Rightarrow \textit{pre}_{C_s} \tag{3'}$$

holds for the input place of $GN_C$, for each constructor $C_s$ $(1 \le s \le p)$.

ii. The implication

$$\textit{logical state of the place } S => Inv \tag{4}$$

holds for each place $S$ of $GN_C$, different from the input place.

iii. For each pair at input and output positions $(R_i, S_j)$ of a transition different from the one realizing the constructor (constructors) of the class, with a transition condition

$$\textit{Member-function is } q \wedge \textit{Condition } P_j \textit{ holds},$$

the Hoare's triple holds:

$$\begin{array}{l} \{ \textit{ logical state of the place } R_i \wedge P_j \} \\ \quad Body_q \\ \{ \textit{ logical state of the place } S_j \} \end{array} \tag{5}$$

For each pair at input and output positions $(R, B_s)$ of the transition, realizing the execution of the constructor (constructors) of the class with a transition condition

$$\textit{Member-function is the constructor } C_s \wedge \textit{Condition } Q_s \textit{ holds},$$

the Hoare's triple holds:

$$\begin{array}{l} \{ Default_C \wedge Q_s \} \\ \quad Body_{Cs} \\ \{ \textit{ logical state of the place } B_s \}. \end{array} \tag{5'}$$

What follows from this definition is that: the places of $GN_C$ present the logical states in which a class object can be; the transitions present the member functions, which can be executed for the current values of the token data, as well as the conditions, under which these member functions can be executed.

Conditions (3) and (3') provide the holding, in the input positions of $GN_C$ transitions, of the preconditions of all member functions which the respective transition executes. Condition (4) ensures the trueness of the class invariant for the current values of the token data in each place of the net, different from the input one. Conditions (5) and (5') ensure keeping the trueness of the predicate *logical state of a place* for the current values of the token data.

Let $GN_C$ is the formal net project of the class $C$. We choose the specification of class $C$ for a specification of $GN_C$.

*Definition 3:* The formal net project $GN_C$ of the class $C$ is correct in respect to its specification if:

a) For transition $T_1$ (Figure 1a) realizing the execution of the class constructor (constructors), the following holds:

For each pair $(R, B_s)$ at input and output positions of the transition with a condition

$$\textit{Member-function is the constructor } C_s \wedge \textit{Condition } Q_s \textit{ holds},$$

a Hoare's triple holds:

$$\{ Default_C \wedge pre_{Cs}(x_{Cs}) \wedge Q_s \} \text{ Body}_{Cs} \{ post_{Cs}(x_{Cs}) \wedge Inv \} \tag{6}$$

Here $x_{Cs}$ is the set of admissible arguments of the constructor $C_s$ and the rest of the denotes correspond to Definition 1 a).

b) For each transition of the type $T_2$ (Figure 1b), different from the transition that realizes the execution of the class constructor (constructors), the following holds:

For each pair ($R'$, $S'$) at input and output positions of $T_2$ with a transition condition

$$\textit{Member-function is } q \wedge \textit{Condition } P' \textit{ holds},$$

a Hoare's triple holds:

$$\begin{aligned} &\{\, pre_{\mathrm{q}}(x_{\mathrm{q}}) \wedge Inv \wedge P' \} \\ &\;\; Body_{\mathrm{q}} \\ &\{\, post_{\mathrm{q}}(\mathrm{x_q}) \wedge Inv \,\}. \end{aligned} \tag{7}$$

Here $x_{\mathrm{q}}$ is the set of admissible arguments of the member function $q$.

Conditions (6) and (7) are respective analogues of conditions (1) and (2) from Definition 1.

**Theorem:** If class $C$ is correct in respect to its specification, the formal GN-project $GN_{\mathrm{C}}$ of the class $C$ is also correct according to its respective specification.
*Proof:* The proof of the theorem is a result of the application of the rule

$$\frac{P \Rightarrow P', \{P'\}\ S\ \{Q'\}, Q' \Rightarrow Q}{\{P\}\ S\ \{Q\}}. \qquad \Box$$

*Definition 4.* The formal GN-project $GN_{\mathrm{C}}$ of the class $C$ corresponds to the realization of the class $C$ if the former is correct in respect to the specification of $C$.

The formal GN-project of a class gives some or all sequences of admissible executions of the class member functions. If it is correct in respect to its specification, it gives some or all sequences of correct executions of the class member functions.

Completion of a transition execution of a formal GN-project of $C$, which is correct in respect to the class $C$ specification, provides a correct execution of the respective member function, as well as the trueness of the predicate *logical state of a place* for the token data in the respective output position. The trueness of the class invariant *Inv*, as well as the pre-condition of each member function whose execution is modeled by the next transition follows from the trueness of *logical state of a place* predicate.

## 3 Conclusion

From the definitions of a formal GN-project of a class and of its correctness, as well as from the theorem as formulated above, it follows that the checking if a GN-model of a class corresponds to the class implementation reduces to: checking if the model is a formal GN-project of the class, e.g. if conditions i), ii) and iii) of Definition 2 hold, as well as if the class realization is correct in respect to the specification defined for it. Building GN-models of OOP classes and researching on these models is an important part of investigation OOP correctness.

The task is additionally complicated in the cases of a large number of OOP classes and when these classes are connected in complex hierarchies. The next step is to define and research the formal GN-projects in these cases, and to study their full compliance with the requirements, [3].

## Acknowledgments

This research is supported by contract 127/2012 of Sofia University Research Fund – 2012.

## References

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

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

[3] Kaloyanova, K., Design from data: how to use requirements for better information system analysis and design, *Proc. of the Int. Conference Informatics in Scientific Knowledge*, Varna, Bulgaria, June 26–29, 2012, 189–197.

[4] Meyer, B., Applying Design by Contract, *IEEE Computer* 25(10), Oct. 1992, pp. 40–51.

[5] Meyer, B., *Object-Oriented Software Construction*, 2nd edition, ISE Inc. Santa Barbara, California, 1997.

[6] 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.