Algorithm of Translation of MSC-specified System into Petri Net

Sergii Kryvyi[1], Lyudmila Matvyeyeva[2]

[1]Politechnical Institute of Rzeszow, Str. Pola, 2, Rzeszow, 35-959, Poland
krivoi@prz.rzeszow.pl, krivoi@i.com.ua
[2]Institute of Cybernetics, NAS of Ukraine, Pr. Glushkowa, 40, Kiev-03187, Ukraine
luda@iss.org.ua

Abstract. We present in this paper the algorithm which performs the translation of MSC'2000 diagrams into Petri net modulo strong bisimulation. The correctness of this algorithm is justified. Here we supplement MSC'2000 standard supplying it with formal definition of MSC element <condition> by means of process algebra. We would mention here that this translation algorithm is the part of the automated verification system which specifies formally and verify designed software or hardware system specified in MSC language, and is presented in [1], and [2].

1 Introduction

While designing and developing of computer systems it is of vital importance to detect and remove defects in product on its early stages in order to avoid time and resource losses. One way of coping with this problem is to use formal methods supported by computer aided verification tools [3], [4]. Actually, formal specification and verification of the software and hardware systems is one of the approaches to improve reliability of the systems under design, to speed up their design process.

Methodologies and toolsets based on formal methods provide the designers with specification languages with formal semantics for a high level description of a software or hardware system and the environment for verification of design choices.

The designers use in their work languages intended for the systems design (as VHDL, MSC, SDL, UML and so on), but verification and analysis of the systems properties require the languages of other kind (languages of mathematical logics, automata theory, algebraic and net languages). A lot of interest exists in developing of automatic interfaces between languages of these kinds. Exactly this problem is resolved in this paper.

2 Problem Statement

Let's consider the algorithm of translation MSC'2000 diagrams into ordinary Petri net modulo strong bisimulation equivalence and the proof of its correctness, which is based on the use of process algebra [5]. The algorithm works over selected subset of basic MSC elements.

We would like to mention here that the algorithm is the part of the verification system which was developed by the authors of this paper, and presented in [1], [2]. The verification system is represented as technological process that applies the formal modeling technique of Petri nets and is based on linear algebra methods of analysis in order to research the properties of a system under design. The verification system is automated and relatively fast compared to other alternatives. It is therefore particularly attractive in industrial context as it can contribute successfully to the reduction of product development costs and production cycle.

The description of the algorithm of translation has been presented in general in [1], also — in [6], and [7]. Detailed description of the algorithm's stages is presented in Sec. 4 below; the Sec. 6 contains the proof of the algorithm correctness.

The most significant feature of the translation algorithm is the way of handling of MSC's element <condition>. This problem in the translation process is quite difficult [8]. In order to solve the problem we have complemented standard semantics of MSC'2000 by adding the formal definition (which we present in Sec. 5 below) of MSC's element <condition>. We would underline that it does not contradict MSC'2000 standard. The process algebra is used to specify formally the element <condition>.

3 Basic Definitions

In this section, we present the basic notion which is used in the paper. We also use well-known definitions of Petri nets theory [9], [10] here.

Definition of strong bisimulation. Let $(S, A, \rightarrow)$ be an LTS — labeled transition system. Where S is a set of states of the system, A is a fixed set of labels, representing actions in the system, $\rightarrow$ — transitions-actions relation in the system. A binary relation $R \subseteq S \times S$ is a strong bisimulation if for all $n_1, n_2 \in S$ such that $(n_1, n_2) \in R$, and any $a \in A$:

– if $n_1 \xrightarrow{a} n_1'$ then $\exists n_2' \colon n_2 \xrightarrow{a} n_2' \wedge (n_1', n_2') \in R$; and

– if $n_2 \xrightarrow{a} n_2'$ then $\exists n_1' \colon n_1 \xrightarrow{a} n_1' \wedge (n_1', n_2') \in R$.

If a strong bisimulation R exists, such that $(n_1, n_2) \in R$, then we say that both states n_1 and n_2 are strongly bisimilar.

4 The Algorithm of Translation of MSC Diagrams into Petri Net

The translation algorithm is presented by its stepwise refinement.

INPUT: A set of the MSC-diagrams, which describe the system under design, in the basic subset of the language MSC'2000 [11], [12].

OUTPUT: A Petri Net (PN) adequate to the set of initial MSC diagrams. The adequacy means that the Petri net generates all those, and only those processes which are feasible for given set of MSC-diagrams.

METHOD: Translation of every MSC diagram of the input set into a Petri Net is performed in two stages.

Stage 1. We build the partial-order graph in order to reflect initial MCS-diagrams events order, which is imposed by static requirements of MSC'2000 standard [12], [11]. The partial ordering is presumed in a minimal form, without an explicit representation of the transitive closure.

Stage 2. We translate the partial-order graph obtained at the stage 1 into the PN.

Reference table is developed while translating process for maintenance of consistent coordination between the input system's descriptions in MSC language and in PN format. This table is necessary to present the results of analysis and verification of the system's formal model in suitable for the system's designer format of MSC-diagrams.

The synthesis of the system from its compound parts takes place in parallel with the translation of each input MSC-diagram. We would mention here that the synthesis principle is based on the use of MSC language element <condition> as the mechanism of synchronization as well as composition of MSC specifications. Detailed description of how synthesis goes has been presented in [1], and [6]. Let's give more detailed description of the algorithm's stages.

Detailed description of stage 1. The partial-order graph consists only from those edges which describe the ordering relation, linking the events of sending and receiving of messages, and the events of setting of the element <condition>. The directed edges of the graph reflect events order which is set by a standard MSC'2000 [11]. We consider only the events of sending and receiving of messages, and setting of the element <condition>. These events of the system correspond to the nodes of the partial-order graph. It is necessary to emphasize, that a <condition> setting is also represented by an event, namely, the event of synchronization. We define the nodes of two types: 1) a graph node which denotes setting of an intermediate element <condition> (not initial and not final <condition> in a diagram) or receiving/sending of a message; and 2) a graph node which denotes setting of an initial or final <condition>. They differ in the way of the translation into PN elements during the stage 2 of the algorithm.

So, the steps of the stage 1 of the algorithm of translation are following.

Step 1. A diagram beforehand assigned as initial (or any diagram from the input set of MSC-diagrams, which we consider as initial) is translated into the graph. Namely, the events of this diagram are translated into the nodes of appropriate types, where the directed edges of the graph reflect the order of these events.

Step 2. Among the remaining diagrams from the initial set of the MSC-diagrams we take one which has the same name of initial and/or final <condition> as accordingly final and/or initial <condition> of the already translated diagram has.

Step 3. **A**. When such a diagram is found, its translation is made according the step 1 and then it is «glued» with already translated part of the graph. Namely, the synthesis from compound parts (corresponding to the initial MSC-diagrams) is carried out. The synthesis passes in accordance with the requirements and the rule represented in [1]. **B**. If such diagram (or diagrams) is not found; then any diagram is taken from diagrams remaining in the input set of MSC-diagrams, and is translated into the graph according the step 1 without «gluing».

Step 4. **A**. If not all diagrams from the input set are translated into the graph, then go to the step 2 with that amendment, that the coincidence with the names of initial and final <conditions> is searched among all diagrams so far translated. **B**. If all diagrams from the input set are translated, then the stage 1 of the algorithm is completed.

Detailed description of the stage 2. We translate the graph, which has been obtained at the stage 1, into the final PN. The initial node of the graph is processed at first. Then subsequent translation of every node and edge of the graph is carried out according the following rules:

– A graph node of the first type (which denotes events of intermediate <conditions> setting and events of a message sending and/or receiving) is translated into a transition of PN.
– Each directed edge of the graph is translated into a place of PN. An arrow of each directed edge of the graph corresponds to an arrow of PN that directs tokens' flow in PN. Thus direction of incoming and outgoing arcs of this PN place corresponds to the orientation of an edge of the graph.
– A graph node of the second type (denoted events of final and initial <conditions> setting) is translated into a transition of PN in the same way as the node of the first type. But important difference exists. Namely, during translation of the node of the second type merging of all places, which correspond to the edges inputting into this node, is carry out; and merging of all places, which correspond to the edges outputting from this node, is carried out also. Fig. 1 illustrates this rule. Thus, p_{in} and p_{out} are places of such merging. t_j is PN transition which corresponds to the node of the second type and models the event of setting of some initial or final <condition> of MSC-diagram.

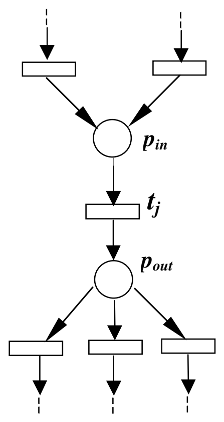

Fig. 1. The translation of the graph's node of the second type

Stage 2 of the translation algorithm is considered as completed, when all nodes and edges of the graph are translated.

5 Formal Semantics of the Element <condition>

Formal definition of standard denotational semantics of the element <condition>, which is represented in the appendix of standard MSC’2000 Annex B [12], needs an enhancement, as standard MSC’2000 [11] and its Annex B [12] determine the element <condition> only as just empty action. In addition, formal definition of composition and decomposition of MSC-diagrams by means of elements <condition> (which is the mechanism of both synchronization and composition of MSC-diagrams) are absent in the standard. These gaps in the standard are filled below in this paper.

In addition to the standard parallel and sequential compositions of the process algebra [13] we introduce two new operators, namely, vertical composition of MSC-diagrams $\otimes$ and horizontal composition of MSC-diagrams $\oplus$ by means of elements <condition>. Namely, axioms Ax1, Ax2, Ax3 are entered.

Ax1: $(x \cdot a) \otimes (b \cdot y) = x \cdot a \cdot y$ if $a \equiv b$, else – $(x \cdot a) \otimes (b \cdot y) = \delta$

Ax2: $(x \cdot a) \oplus (y \cdot b) = (x \| y) \cdot a$ if $a \equiv b$, else – $(x \cdot a) \oplus (y \cdot b) = \delta$

Ax3: $(a \cdot x) \oplus (b \cdot y) = a \cdot (x \| y)$ if $a \equiv b$, else – $(a \cdot x) \oplus (b \cdot y) = \delta$

where a and b are an atomic actions of the process algebra. They correspond to the events of setting of <condition>, which is shared, and by means of which diagrams are "glued". x and y are terms of process algebra. These terms determine semantics of those and only those instances which share the element <condition>, by means of which diagrams are "glued".

These new introduced compositions represent formally «gluing» of MSC-diagrams by means of the operator of the so-called strict sequential composition •, and by the operator $\|$ of parallel composition. It is significant, that the operator of weak sequential composition $\circ$, which is used in Annex B [12], does not take into acthe graph the obvious mechanism of synchronization and therefore is not considered here.

Also we introduce the definition of new operator λ_S which states, that element <condition> is the means of synchronization of the events. The static requirements of the standard MSC’2000 [11] describe this statement as following.

– If two instances share the same <condition> then for each message exchanged between these instances, both sending of message and its receiving must be placed either both before or both after setting of <condition>;
– If two conditions are ordered directly (because they have an instance in common) or ordered indirectly via conditions on other instances, this order must be respected on all instances that share these two conditions.

We would mention here that the general idea of the semantics of a MSC-diagram is that it is the free merge of its constituent instances [13]:

$S[ch] = \lambda(\|_{i\in Inst(ch)} S_{inst}[i])$, where S – semantics of MSC-diagram, ch — a MSC-diagram, S_{inst} — semantics of one instance, i — an instance, λ — a state operator. Interleavings, in which a message output is preceded by its corresponding message input, are enabled by this construction. An operator λ enables only those interleavings that respect the constraint above and is introduced in axiomatic way in [13]. Let us consider for example elementary diagram *msc_TwoInstances* (see Fig. 2). We have two instances exchanging one message. A message is divided into two parts: a message output and a message input. The semantics of instance *i* is *out(i,j,k)*. The semantics of instance *j* is *in(i,j,k)*. The basic idea is that the two instances operate in parallel, independently of each other. The operator λ enforces the basic static requirement that a message must be sent before it is received. So, the semantics of an MSC-diagram is derived from the semantics of its instances by placing them in parallel and removing unwanted execution orders by applying the operator λ .

The semantics of the diagram *msc_TwoInstances* is as:

$$\lambda(out(i,j,k)\|in(i,j,k)) =$$
$$= \lambda(out(i,j,k)\cdot in(i,j,k) + in(i,j,k)\cdot out(i,j,k)) = out(i,j,k)\cdot in(i,j,k),$$

where *out(i,j,k)* means sending of message *k* to instance *j*, and *in(i,j,k)* — receiving message *k* from instance *i*.

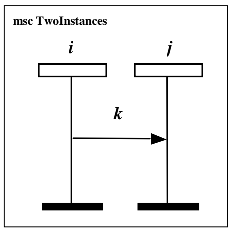

Fig. 2. Elementary MSC-diagram

The definition of new static operator λ_S is represented in a table 1. The semantic function for MSC-diagram *ch* is defined by: $S[ch] = \lambda_S(\lambda(\|_{i\in Inst(ch)} S_{inst}[i]))$, where operator λ_S is used after application of operator λ defined in [13].

Table 1. Formal semantics of the operator λ_S

LS1: $\lambda_S(\varepsilon) = \varepsilon$	
LS2: $\lambda_S(\delta) = \delta$	
LS3: $\lambda_S(x+y) = \lambda_S(x) + \lambda_S(y)$	

LS4: $\lambda_S(a \cdot x) = \lambda_S(x)$	$a \in A_C \wedge x = a \cdot y$
LS5: $\lambda_S(a \cdot x) = a \cdot \lambda_S(x)$	$(a \notin A_C \vee (a \in A_C \wedge x \neq a \cdot y)) \wedge before_a(x)$
LS6: $\lambda_S(a \cdot x) = \delta$	$(a \in A_C \wedge ((mes, i, j) \in M, a = condition\ (a, i, j))) \vee \neg before_a(x)$

Let us explain the axioms LS1 – LS6 which define a static operator λ_S. a is atomic action, x and y are the terms of the process algebra [13]. A_C means the set of the elements <condition>, which are set in a MSC-diagram, M is the set of names of sent, but so far not received messages. After the execution of the message input the message name is removed from the set M. The expression *condition(a,i,j)* means setting of <condition> a, which refers to the instances i and j (or <condition> covers the instances i and j). The mapping $before_a(x)$ is defined in the standard of MSC'2000 [12] and means that event a takes place before any event, which is contained in the term x. Axiom LS4 states that the operator λ_S merges together a few copies of the same element <condition>, which are sequentially composed (i.e., idempotency of the element < condition >). If a is the event of sending/receiving of a message or setting of <condition>, which can not be «glued» according the axiom LS4, and all events composing the term x follow later on an instance, on which an event a happened, then we execute a and further apply an operator λ_S to the term x (LS5). If exchanging message *mes* between the instances i and j not happened either both before or both after setting of <condition> which covers these instances (see Fig. 3), then such branch of process is deadlocked (LS6). The axiom LS6 states also that if in the sequence of mutual interleaving of events the violation of the static requirement of the standard MSC'2000 (to keep strict order of events along all instances) takes place then such branch of process is deadlocked.

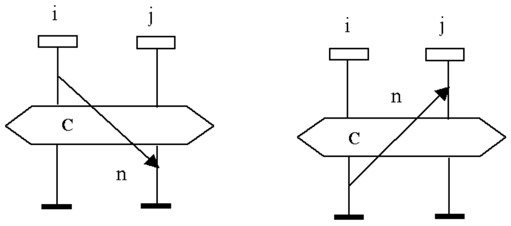

Fig. 3. Violation of the semantics of an element <condition>

Thus, in this section we have extended the semantics of MSC by adding a formal algebraic semantics of the MSC's element <condition> via static operator λ_S, and two new operators, namely, vertical composition of MSC-diagrams $\otimes$ and horizontal composition of MSC-diagrams $\oplus$. These new operators are used for justification of the algorithm of translation in the next section, and do not contradict standard semantics of MSC [11], [12] but just enhance it.

6 The Theorem and the Proof

The theorem, which justifies correctness of the algorithm of translation, is the following.

Theorem. Initial MSC-diagrams and synthesized by these diagrams Petri net, which is built in accordance with the algorithm of translation, are (strong) bisimilar equivalent.

Proof. The proof scheme is following. We build an algebraic expression by input MSC-diagrams. For that we use formal semantics of MSC, namely, both standard one [12], [13] and new operators introduced in Sec. 5. According to standard semantics of MSC every diagram of the language is a parallel composition of algebraic expressions representing semantics of its composing instances. The unfolding of parallel composition undergoes by means of axioms (see [12], [13] and Sec.5). After that in the obtained algebraic expression the impossible sequences of events are deleted via the application of state operator λ , and then — state operator λ_S .

The obtained algebraic expression (or compound term) determines the total-ordered interleaving of its subterms, which represent events on instances of MSC-diagrams. In other words, we build a transition system according to given algebraic expression. Thus, the algebraic expression, which is obtained as a result of application of the axioms, represents the set of all possible traces of the system.

Such set of all possible traces of the system must be equal to the set of all possible sequences of firings of transitions of PN, which is received from the input MSC-diagrams by application of the algorithm of translation. This means the bisimulation equivalence of input MSC-diagrams and synthesized PN. In other words, we build a transition system according to this PN, and prove the bisimulation equivalence of obtained transition systems by the method of mathematical induction on the number i of events in the system. Every induction step covers one event of the system.

(The basis of induction i=1). Trivially true for traces of length one.

(The step of induction). Let the set of all possible traces of the system consists of i events, and is equal to the set of all possible sequences of firing of transitions of PN, which has i transitions. Consider (i+1)-th event ev_i. Formal semantics of process with the added (i+1)-th event ev_{i+1} looks like algebraic expression which represents the new set of all possible traces. We must prove that it must be equal to the set of all possible traces (or sequences of firings of transitions) of output PN with the added (i +1)-th transition t_{i+1}. Here, each pair of events can be in one of the tree relations: ordering $ev_i \prec ev_{i+1}$, conflict $ev_i \# ev_{i+1}$, and concurrency $ev_i \parallel ev_{i+1}$.

By contradiction assume that there is a trace including (i +1)-th event ev_{i+1} of the MSC system, and there is no PN trace (i.e. a sequence of firings of transitions) having the corresponding transition t_{i+1} in (i +1)-th position.

1) Consider **sequential** execution of i -th and (i +1)-th events, i.e. ev_i is a direct predecessor of ev_{i+1}: $ev_i \prec ev_{i+1}$ (the relation $\prec$ is the strong ordering relation; it also is irreflexive transitive closure of incidence relation between places and transitions of PN). In other words, let us prove if not $ev_i \prec ev_{i+1} \Leftrightarrow t_i \prec t_{i+1}$. It could happen if there is a pair of ordered events $ev_i \prec ev_{i+1}$ in trace for which there is no path from t_i to t_{i+1}, or there is a pair of transitions t_i and t_{i+1}: $t_i \prec t_{i+1}$, and ev_i is not ordered before ev_{i+1}.

Both assumptions contradict the one-to-one correspondence between the strong ordering relations of events in MSC-diagram and transitions in appropriate PN.

Indeed, $ev_i \prec ev_{i+1}$ means that either the event ev_i is on one instance with the event ev_{i+1} (i.e., the events belong to the same process/instance), or they must be respectively the send and receive events of the same message. Then the assertion «transition of t_i is connected by a way with transition t_{i+1}» is true by construction according the algorithm of translation. That is, either the sequential path of transitions of PN or the construction of synchronization (see Fig. 4) is gotten as a result of the translation. As a result of the translation the construction of merging (see Fig. 5) is also possible in the case of multiple «gluing » of MSC-diagrams.

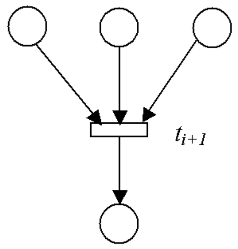

Fig. 4. Synchronization construction

Such «gluing» is expressed formally as vertical and horizontal compositions of MSC-diagrams by means of the element <condition>, which are defined by means of the axioms AX1,AX2, and AX3 (in previous section).

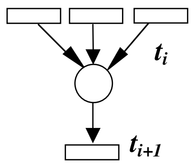

Fig. 5. Merging construction

2) Consider **concurrent** execution of i -th and (i +1)-th events (it means that the events can take place either in any order or simultaneously).

Let us prove if not $ev_i \parallel ev_{i+1} \Leftrightarrow t_i \parallel t_{i+1}$. It could happen if there is a pair of concurrent events: $ev_i \parallel ev_{i+1}$, and the corresponding transitions are not concurrent, or there is a pair of transitions t_i and t_{i+1}: $t_i \parallel t_{i+1}$, and ev_i is not concurrent regarding ev_{i+1}. Both assumptions contradict the one-to-one correspondence between the concurrency relations of events in MSC-diagram and transitions in appropriate PN.

Indeed, it is obvious by construction. $ev_i \parallel ev_{i+1}$ means that these events take place on different instances and are not send-receive pair of a message, so they are asynchronous, and from the point of view of total-ordered interleaving semantics there is the non-deterministic choice between these two events (see process algebra basic axioms in [13]). During the stage 1 of the algorithm of translation the node, which corresponds to the event ev_{i+1}, is created. The constructed chain of edges of the graph has one general node of the *first type*, appropriate to the event ev_k, which precedes the nodes appropriate both ev_{i+1} and ev_i. The concurrency construction (see Fig. 6) is gotten as a result of the translation from the graph into PN. Thus, transitions of t_i and t_{i+1} are asynchronous and, consequently, following is executed: $t_i \parallel t_{i+1}$.

3) Consider now when i -th and (i +1)-th events are in **conflict**: $ev_i \# ev_{i+1}$, i.e. execution of one event eliminates opportunity of other event to be executed.

Let us prove if not $ev_i \# ev_{i+1} \Leftrightarrow t_i \# t_{i+1}$. It could happen if there is a pair of events in conflict and the corresponding transitions are not in conflict, or there is a pair of transitions t_i and t_{i+1}: $t_i \# t_{i+1}$, and corresponding events ev_i and ev_{i+1} are not in conflict. Both assumptions contradict the one-to-one correspondence between the conflict relations of events in MSC-diagram and transitions in appropriate PN.

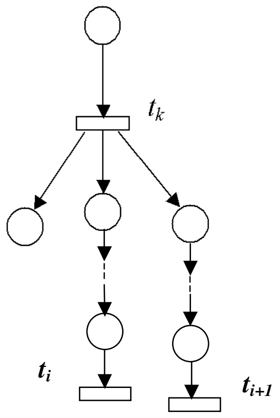

Fig. 6. Concurrency construction

Indeed, consider $ev_i \# ev_{i+1}$. It means that i-th and (i +1)-th events take place in the different MSC-diagrams connected by non-deterministic alternative composition (see the rule of the synthesis of MSC-diagrams in [1]). By other words, multiple «gluing» of these diagrams by an element <condition> has happened. Moreover, the events ev_i and ev_{i+1} take place on the instances covered by one element <condition>, by which multiple «gluing» takes place. The event ev_i represents a part of the process set by one diagram, and the event ev_{i+1} it is a part of the process set by other diagram. Consider the event of setting of given <condition> by which multiple «gluing» of these diagrams takes place. It is the predecessor of the events ev_i and ev_{i+1}. During the translation of this event of setting of <condition> on the stage 1 of the algorithm of translation the node of the *second type* is created. On the stage 2 of the algorithm this node is translated into PN transition, where all its output places are merged together in one place: P_{choice} (see Fig. 7).

Thus, the construction of conflict of PN is received after completion of stage 2 of the algorithm. With respect to said above following is executed: $t_i \# t_{i+1}$, i.e. there are two transitions, which are in conflict. It corresponds to choice construct shown in Fig.7, and models alternative composition between two MSCs. Thus, it is proved that $ev_i \# ev_{i+1} \Leftrightarrow t_i \# t_{i+1}$.

So, all possible variants of order of events in the system are considered, and it is proved that the set of all possible traces of MSC system is equal to the set of all possible traces (i.e. the sequences of firings of transitions) of PN, which is received from the initial set of MSC-diagrams by application of the algorithm of translation. It means strong bisimulation equivalence of initial MSC-diagrams and synthesized PN. The theorem is proved.

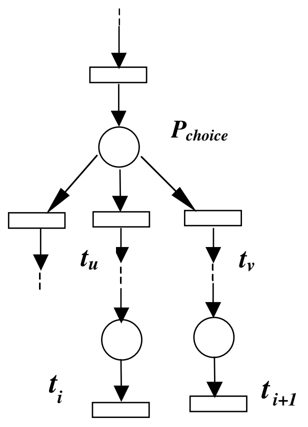

Fig. 7. Conflict (or choice) construction

7 Conclusions

Thus, let us conclude that the most significant feature of the given algorithm is the way of handling of MSC's element <conditions>. The important result is the enhancement of MSC'2000 standard by the formal definition of the element <condition> semantics.

The translation algorithm, presented in the paper, considers only the subset of MSC language, so extending of this subset is a direction of the further research. And what is more, an important area of study is obtaining of necessary experience so that to develop translators and toolsets for the languages: SDL, UML, etc.

References

1. Kryvyy S., Matvyeyeva L., Lopatina M. Automatic Modeling and Analysis of MSC-specified Systems. Fundamenta Informaticae, Vol.67, N. 1 – 3, August 2005, pp.107–120
2. Kryvyy S., Matvyeyeva L., Lopatina M. Automatic Analysis ans Verification of MSC-specified Telecommunication System. ICINCO'2004 August (25-28) in Setúbal/Portugal. In Proceedings of First International Conference on Informatics in Control, Automation and Robotics, Vol.2, pp. 402-405

3. Adriaan de Groot, Hooman J., Kordon F., et al. A Survey: Applying Formal Methods to a Software Intensive. In Proceedings of the IEEE High-Assurance Systems Engineering Workshop, 2001, pp. 55 - 64
4. Clarke E., Wing J.: Formal Methods: State of the Art and Future Directions, CMU Computer Science Technical Message CMU-CS-96-178, August 1996. Published in: ACM Computing Surveys, Vol. 28, N. 4, 1996, pp. 626-643
5. Bergstra, J.A., Klop, J.W.: Process Algebra for Synchronous Communication. Inf.&Control, Vol. 60, 1984, pp.109–137
6. Kryvyy S., Matvyeyeva L., Lopatina M. Automatic Translation of MSC Diagrams into Petri Nets, International Journal “Information Theories & Applications”, Vol.10, 2003, N.4, pp.423-430
7. Kryvyy S., Matvyeyeva L., Lopatina, M.: Automatic Transformation of MSC Diagrams into Petri Nets. In Proceedings of SCI’2003, Orlando, USA, 29-31 July 2003, Vol. 5, pp. 140–146
8. Mauw S., Reniers M.A.: Thoughts on the meaning of conditions. Experts meeting SG10, St.Petersburg TD9016, ITU-TS, 1995
9. Peterson J. L.: Petri Net Theory and the Modeling of Systems. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1981
10. Murata T.: Petri Nets: Properties, Analysis and Applications. In Proceedings of the IEEE, Vol.77, N. 4, 1989, pp. 541–580
11. ITU-TS Recommendation Z.120: Message Sequence Chart (MSC). ITU-TS, Geneva, 2000
12. ITU-TS Recommendation Z.120 Annex B: Algebraic semantics of Message Sequence Charts. ITU-TS, Geneva, 1995
13. Mauw S., Reniers M.A.: An algebraic semantics of Message Sequence Charts, The Computer Journal, Vol.37, N.4, pp.269-277, 1994
14. Kluge O.: Time in Message Sequence Charts Specifications and How to Derive Stochastic Petri Nets. In Proceedings of the Third International Workshop on Communication Based Systems (CBS3), Berlin, March 2000
15. Heymer S.: A Semantic for MSC based on Petri-Net Components. SIIM Technical Message A-00-12, June 2000. Informatik-Berichte Humboldt-Universität zu Berlin, 2000
16. Kluge, O., Padberg, J., Ehrig, H.: Modeling Train Control Systems: From Message Sequence Charts to Petri Nets. Technische Universität Berlin. Retrieved on April 20, 2001 from http://cs.tu-berlin.de/SPP/index.html
17. Grabowski J., Graubmann P., Rudolph E.: Towards a Petri Net Based Semantics Definition for Message Sequence Charts. In SDL’93 Using Objects, Proceedings of the 6th SDL Forum, Darmstadt. O.Fergemand and A.Sarma (eds.), Elsevier Science Publishers B.V., 1993
18. Esparza J., Melzer S.: Verification of Safety Properties Using Integer Programming: Beyond the State Equation. Formal Methods in System Design, N. 16, 2000, pp. 159–189
19. Clark E.M., Grumberg O., Peled D.: Model Checking. MIT Press, 1999
20. Pixley C.: Formal Verification 2004. EDA Tools Forum, 2004
21. Holzmann G., Smith M.: Automating Software Feature Verification. Bell Labs Technical Journal, Vol. 5, 2000, pp.72-87
22. Mäkelä M. Maria: Modular Reachability Analyser for Algebraic System Nets. ICATPN 2002, LNCS, Vol. 2360, pp. 434-444. J. Esparza and C. Lakos (eds.), Spinger-Verlag Berlin Heidelberg 2002
23. J.-M. Colom and M.Koutny (eds.). Application and Theory of Petri Nets 2001, Proceedings of the 22-nd International Conference, ICATPN 2001, Newcastle upon Tyne, UK, June 25-29, 2001, LNCS, Vol. 2075. Spinger-Verlag 2001