PCX: A Translation Tool from PROMELA/Spin to the C-Based Stochastic Petri Net Language

Abstract: Stochastic Petri Nets (SPNs) are a graphical tool for the formal description of systems with the features of concurrency, synchronization, mutual exclusion and conflict. SPN models can be described with an input language called CSPL (C-based SPN language). Spin is a generic verification system that supports the design and verification of software systems. PROMELA (Protocol or Process Meta Language) is Spin's input language. This work provides the translation rules from a subset of PROMELA constructs to CSPL, and also offers an experimental tool PCX (PROMELA to CSPL Translator) and approach to explore the specification and analysis of stochastic properties for systems. The PCX tool translates the formal description, written in PROMELA, into an SPN, represented by CSPL. The approach requires users to add stochastic property information, during (or after) the translation. Translation of the PROMELA model to a CSPL specification will allow the analysis of non-functional requirements such as reliability, availability, and performance through SPNP (Stochastic Petri Net Package), a stochastic analysis tool. This is useful in the design and validation of performance where parameters such as failure rate or throughput are available. Moreover, certain structural and architectural features of software can be evaluated and considered within the context of Spin-verifiable properties. This approach provides additional flexibility to the PROMELA specification-modeling paradigm to include stochastic analysis of structural and non-functional properties. Thus, PCX provides a practical bridge between system verification and system validation. **Keywords:** Petri Nets, Spin, PROMELA, translation tool, verification and validation analysis

1 Introduction

Ideally verification and validation of a software design specification would be possible before any code was generated. Indeed, in a perfect world we would know that the implementation was correct because we could trust the class libraries, the development tools, verification tools and simulations etc. These features would provide the confidence needed to know that all aspects (complexity, logical and timing correctness) of the design were fully satisfied (i.e., everything was right). Right in the sense that we built it right (its correct with respect to its specification) and it solves the right problem. Unfortunately, our ability to ensure the correctness has not kept pace with the growth in system complexity [1].

Although verification and validation methods have fallen behind the increasing complexity of new and evolving systems, there have been several developments that may close the gap. One of these developments has been in the field of formal methods. Such methods, typically given by a

formal specification language, provide frameworks within which people can specify, develop, and verify (and/or validate) systems in a systematic manner. These techniques use mathematical logics to explore the state space of complex systems and to verify their correctness against carefully stated correctness properties or to derive analytic performance measures [2-4]. These properties/measures can be either functional requirements (communications, control, redundancies) or non-functional requirements (performance, reliability, execution deadlines). Tools based on these techniques can be used to show that systems will behave as expected for all possible cases. Naturally the precision of such results depends on how closely the model represents the actual system under study (e.g., an existing or proposed system).

1.1 Petri nets are used to assess non-functional characteristics

Petri Nets (PNs) are a graphical tool for validating the formal description of systems (typically distributed systems) that possess the characteristics of concurrency, synchronization, mutual exclusion and conflict [5]. The stochastic Petri Net (SPNs) is a PN augmented with stochastic attributes, such as a rate or probability of a transition firing. Techniques that utilize SPNs are good for evaluating the performability, but they may be too abstract and cumbersome from the standpoint of specifying and evaluating functional behavior. Therefore, one major objective of this work is to provide an integrated approach to assist the user in specifying both functional and non-functional requirements.

1.2 PROMELA/Spin used to assess logical functional properties

Spin is a generic system that supports the design and verification of distributed software systems. Spin verification models are focused on proving the correctness of process interactions, and they attempt to abstract as many states as possible from internal sequential computation [6]. PROMELA

(Protocol or Process Meta Language) is the input language for Spin [7]. This work provides translation rules from PROMELA to Stochastic Petri Nets, and also offers an experimental tool PCX (PROMELA to CSPL Translator) and approach codified by the PCX tool to explore the specification and analysis of stochastic properties. In this way, the merits of a powerful modeling technique for performability analysis (using SPNs) can be combined with a well-defined formal specification language (PROMELA) for logical analysis (i.e. verification). By doing this, we can come closer to providing a formal approach to designing a functionally correct system that meets the reliability and performance goals [8, 9]. In the following sections we provide a brief description of the overall PCX translation strategy and related work. We address the question of just how faithful (i.e., equivalent) are the different formalisms in terms of their underlying state space (i.e., comparing the state space generated for the same model represented using both PROMELA and SPNs). Section 3 describes the implementation of PCX. Section 4 and 5 present the experimental results, conclusions and future research directions respectively. The appendix gives the basic set of canonical translations.

2 Translation tool from PROMELA to CSPL

The PROMELA/Spin to PN translations were designed to facilitate automatic decomposition of the PROMELA/Spin model (construct-by-construct) into PN sub-components, and then the sub-components are linked together to form a complete system Petri Nets.

2.1 Related work

The PCX tool abstracts the control flow (i.e., structural characteristics and dynamic aspects) from the PROMELA model and translates the flow control into a Stochastic Petri Net model, represented using CSPL. During or after the translation, the tool allows users to add stochastic properties that the

PROMELA models do not provide.

Holzmann presented an approach for the translation of Petri Nets into a PROMELA model [10]. The approach uses the idea that Petri Nets can easily be represented with a small subset of PROMELA constructs. Grahlmann has developed the PEP tool (Programming Environment based on Petri Net) that incorporates a feature that translates PNs into PROMELA for analysis using Spin based on the same idea [11]. In their approach the resulting PROMELA models have the same state space as the Petri Net model does. However, their method only translates from Petri Nets (i.e., not Stochastic Petri Nets) to PROMELA. The state space is the same for the translation from PNs to PROMELA using the PEP tool. Using the PCX tool translation, the state space of PROMELA model is different from that of the PN model because the PCX translation rules utilize a simple abstraction method. Intuitively, the expressive power of PROMELA is much greater than that of PNs (which only have places. transitions, arcs and tokens). This constitutes the major reason for the difference.

The general approach used for the PCX tool is abstraction. PCX captures the control flow of the PROMELA input language as the basis for the non-functional structural (stochastic) analysis. Sheldon presented the CSPN tool (Communicating Sequential Processes (CSP): CSP-to-Stochastic Petri Nets) that enables designers to investigate functional and non-functional requirements by translating CSP to Stochastic Petri Net while assigning stochastic properties [12]. From a methodological point of view, our work was mainly inspired by the experience of the CSPN tool. The main differences between PCX and CSPN are the input language. The back-end part of the PCX tool is similar to that of the CSPN tool.

2.2 Methodology

This work offers an approach to verify system correctness (against logical assertions or systems requirements) using model checking and validate system behavior (reliability and performance)

based primarily on the structural characteristics of a formal specification. The PCX tool provides a notion of refinement that allows the designers to describe a system at an appropriate abstract level.

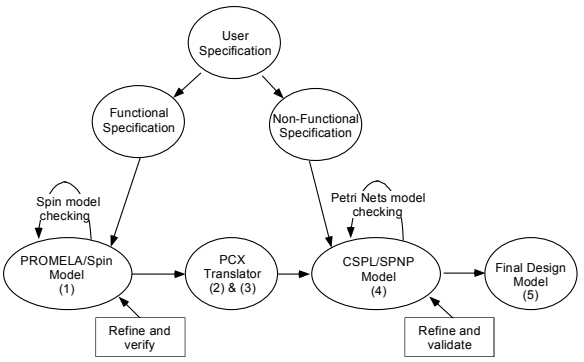

Figure 1. The PCX approach for the system model analysis

Figure 1 shows how this approach involves abstraction from the requirements specification into a design specification and then further evaluations based on stochastic analysis of the system models. First the user develops the PROMELA model based on the requirement specification that can be verified by the Spin model checker. Then the verified model, written in PROMELA, is translated into SPN model, written in CSPL. During or after this translation, the PCX tool allows users to add stochastic parameters that can be based on the system requirement specification. The user analyzes the SPN model for stochastic properties using SPNP to obtain the final design model. If the final design model does not meet both functional and non-functional requirements, the above steps are repeated until the user requirements are satisfied. The PCX tool is used in the context of the 5 steps listed in Table 1. Each step in Table 1 also is marked in Figure 1 with same step number.

Table 1. Methodology: steps for specification and analysis

Step	Description of steps used in the approach
1	PROMELA/Spin model for analyzing the logical consistency
2	Translate from PROMELA to Stochastic Petri Nets.
3	Assign performance and reliability parameters among subsystem components.
4	Analyze the SPN for stochastic properties [using SPNP] (validate performance and reliability goals using stochastic system models).
5	Final design based on results from Spin and SPNP tool analysis

The PROMELA-to-SPN translation rules used for process decomposition and composition are codified in the PCX tool. PCX decomposes individual PROMELA constructions into PN structures. The elemental structures are linked together in a hierarchical fashion according to their adjacency and nesting within the PROMELA specification. Having created this net of linked structures, PCX traverses and expands sub-SPNs into the complete system described by a PN. Also, while PCX decomposes the PROMELA constructions, the service and failure rate annotations are added via user inputs for being incorporated later into the CSPL specification file. After the preliminary structure of the SPN is complete, PCX must reconcile synchronization points because the PROMELA message channel must rendezvous at a particular point in which the message is sent/received. PCX finally generates an SPN graphic specification file and a CSPL specification file. The user can view the SPN's distribution of places and transitions as a graph after the translation is accomplished. The resultant PN models (CSPL file) with the different stochastic parameters can be analyzed using the SPNP tool. This entire process occurs at various levels of user controllable interaction. In essence, the approach provides for systematic and automatic translation and subsequent augmentation (e.g., failure rates, service rates, and deadlines) of the PROMELA model into an SPN model for evaluating both functional and non-functional properties.

2.3 Mapping from PROMELA to SPN

An initial set of translation rules from the PROMELA specification language into the SPN was defined by Chuck Rodacker [13]. The general principle behind the translation is the following. In PROMELA, each statement can be viewed as a condition (or event), which are represented by a place. After execution of one statement, another statement can execute, while these actions themselves are viewed as transitions (i.e., SPNs provide the mechanism to analyze the system performance based in structural characteristics). Therefore it is appropriate to abstract control flow

information from the PROMELA model. The control flow then provides the basis for analyzing performance. This is the essential information that is needed and PCX captures the control flow information of an entire PROMELA specified model. This translation approach is similar to the methods in [11]. Moreover, PCX focuses on abstracting control flow information and therefore the content of a statement is not needed in the resulting SPN model. PCX allows users to assign rates to each timed transition for the basis of performability analysis, otherwise a default value of 1.0 is assigned.

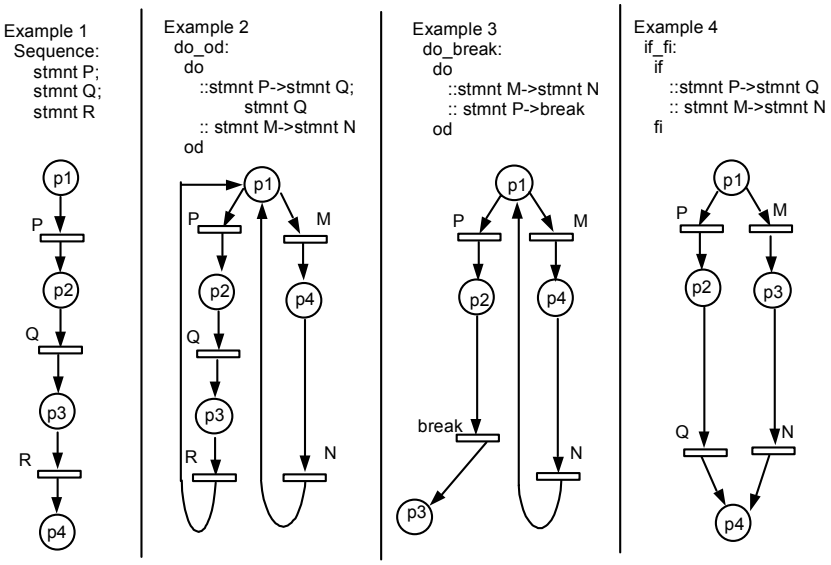

Figure 2. The examples of the translation from PROMELA to PN

PCX allows the user to set probability parameters to PROMELA models (such as if_fi

```
short state1 = 1,  state2=10;
proctype A()
{
  state2=state2-state1;
  if
    ::state2=state2+state1
    ::state2=state2-state1
  fi;
  state2=state2-state1
}
init
{
  run A()
}
```

The set of reachable states for the if-fi construct

State	Token in place
M0	P00
M1	P01
M2	P02
M3	P03
M4	P04
M5	P05
M6	P06

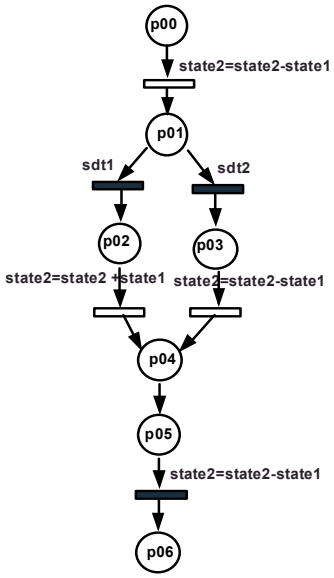

Figure 3. The if-fi translation example

constructs, do_od constructs). In Example 4 of Figure 2, PCX assigns stochastic rate parameters for transition P and transition Q which in turn determines the ratio of time a token is flowed to either p2 or p3.

The PROMELA to SPN translation is designed to facilitate automatic decomposition of the PROMELA constructs into SPN sub-components and their subsequent compositions into a complete system SPN. Some example translations are given in Figure 2.

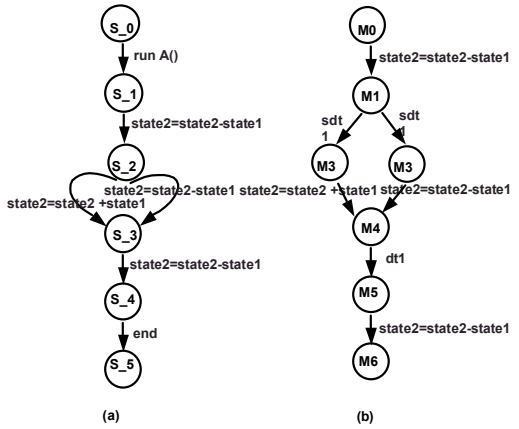

Figure 4. The reachability graph of the if-fi example

3 Results

Example A contains an if_fi construct. There is one statement each before and after this construct. The resulting SPN shows that a conflict transition occurs from place p01. For conflict transitions, the PCX tool can allow users to set probability parameters (or use default values). The parallel part of the SPN model (from places p01 to p04) represents if_fi construct in PROMELA. The light bars in Figure 3 represent timed transitions for which rates must be assigned[1]. There are 7 reachability states M0, M1, M2, M3, M4, M5 and M6 when a token is in p00, p01, p02, p03, p04, p05, p06 respectively. The reachability graph of the resulting SPN model is shown in Figure 4(b). Compared with the reachablity graph obtained directly from the PROMELA model shown in Figure 4(a), the state S_0 which is initialized in Figure 4(a) for run A( ), can be combined with S_1 because run A( ) does not provide a new state in this case. Also, the state S_5, S_6 in Figure 4(b) that represent the end process, do not provide new states. During the translation, sdt1 and sdt2 are provided for adding probability parameters, so the state M2 and state M3 in Figure 4(b) are added compared with the

[1] Again, the PCX tool allows users to set these rates (or use default values).

```
short state1 = 1, state2=10;
proctype A()
{
state2=state2-state1
do
::state2=state2+state1;
   state2=state2+state1
::state2=state2-state1;
   state2=state2-state1;
   state2=state2-state1
   if
     state2=state2-state1
     state2=state2+state1
   fi
od
}
init
{
run A()
}
```

The set of reachable states for the do-od construct

State	Token in place
M0	P00
M1	P01
M2	P02
M3	P03
M4	P04
M5	P05
M6	P06
M7	P07
M8	P08
M9	P09
M10	P10
M11	P11

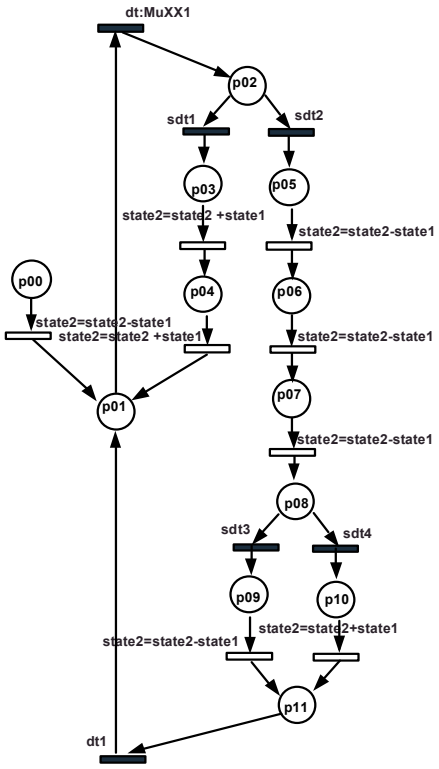

Figure 5. The do_od translation example

reachability graph in Figure 4(a).

In example B we see the if_fi, do_od, and sequential construct used. The resulting SPN is shown in Figure 5. The SPN model displays a sequential operation (from state p00 to state p11), a parallel operation (from place p02 to place p11, and from place p8 to place p11), and a repeating loop (from place p01, then p02… then to place p01). There are 12 reachability states M0, M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11 which corresponds to a token is in places p00, p01, p02, p03, p04, p05, p06, p07, p08, p09, p10, p11 respectively.

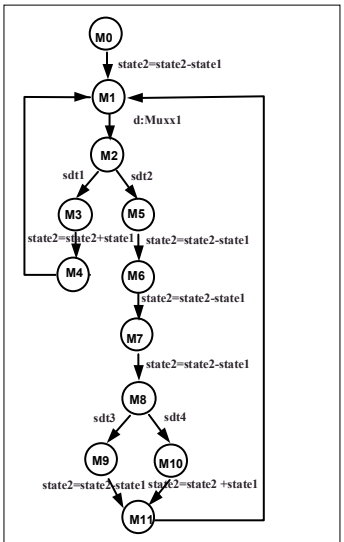

Figure 6. The SPN model of the do_od construct

4 Conclusion and future study

The objective of this work was to show the feasibility of translating

PROMELA into SPNs for the purpose of reliability and performability analysis. Such translations can give insight (1) into the feasibility of meeting non-functional requirements, (2) help to identify the best candidate design based on a formal description of the system, and (3) help to identify failure modes and fault handling mechanisms. This approach enables the stochastic properties of the system specification to be ascertained while allowing the parameters used in the analysis to be formally captured in the PROMELA/Spin model. Subsequent analyses can be run without having to rewrite all of the pertinent values. Only those parameters that are identified as critical in terms of their impact to the integrity of the overall system (i.e., sensitivity analysis) need be perturbed.

PCX combines the power of two other tools namely dot (for viewing the graphical PN representation) and SPNP. PCX offers a selection of command line options. Most of PCX's current features are driven by the SPNP functionality. An interactive menu is used to control run parameters related to the type of analysis (e.g., precision, iterations, generating a reachability graph, running continuous time versus discrete time Markov analysis, etc.). PCX also allows the designer to parameterize and control the characteristics of the system under study (e.g., setting priorities, rates or probabilities among transitions, etc.). In general, the PCX tool provides a new level of abstraction and basis for understanding interactive concurrent process algebraic specifications by leveraging the power of dot and SPNP.

In the future, it will be necessary to define additional PROMELA/PN canonical translation rules for a larger set of the PROMELA language. In addition, we plan to develop more examples that demonstrate the usefulness of this approach.

5 References

1. Pressman, R.S., Software Engineering: A Practitioner's Approach. Fourth ed. 1997: The McGraw-Hill Companies, Inc.

2. Craigen, D. Formal Methods Adoption: What's working, What's not! in The 6th International SPIN Workshop on Practical Aspects of Model Checking. 1999. Toulouse, France.
3. Gerhart, S.L., Application of Formal Method: Developing Virtuous Software. IEEE Software, 1990(Sept. 1990): p. 7-10.
4. Reisig, W. Combining Petri Nets and Other Formal Methods. in 13th International Conference on Application and Theory of Petri Nets. 1992.
5. Marsan, M.A., Balbom, G., Gonte, G., Donatelli, S., Franceschinis, G., Modeling With Generalised Stochastic Petri Nets. 1995, New York, NY 10158, USA: John Wiley And Sons.
6. Holzmann, G.J., The Model Checker SPIN. IEEE Transactions on Software Engineering, 1997: p. 279-295.
7. Holzmann, G.J., Design and Validation of Protocols: A tutorial, in Computer Networks and ISDN Systems. p. 981-1017.
8. Wang, C., Trivedi, K.S., Integration of specification for Modeling and Specification for System Design. 1994.
9. Wang, C., Some Problems in the Specification and Analysis of Computers and Networks. 1995.
10. Holzmann, G.J., SPIN Verification Examples and Exercises., http://cm.bell-labs.com/cm/cs/wat/spin/Man/Exercises.html.
11. Grahlmann, B. and C. Pohl. Profiting from Spin in PEP. in The 4th International SPIN Workshop. 1998. Paris, France.
12. Sheldon, F.T., Specification and Analysis of Stochastic Properties for Concurrent Systems using CSP, in Department of Computer Science and Engineering. 1996, The University of Texas at Arlington: Arlington, Texas.
13. Rodacker, C., PCX- A Translation Tool, Master Project, Unpublished Report, in Department of Computer Science. 1999, University of Colorado Springs: Colorado Springs.

Appendix A: PROMELA to SPN translation rules

The collection of the standard translations from PROMELA to SPNs is provided here. Users can combine the basic units to form the actual model.

The Sequence Construct

The statement stmnt P, Q, R in PROMELA represents three *transitions* (transition P, Q and R shown in SPN model). The sequential execution flows from one place (e.g., p1 shown in the SPN model is assigned with one character and serial number by the PCX tool during the translation) to another place (e.g., p2 shown in the SPN model). In this case, the state space in PROMELA model is the same as that in SPN model.

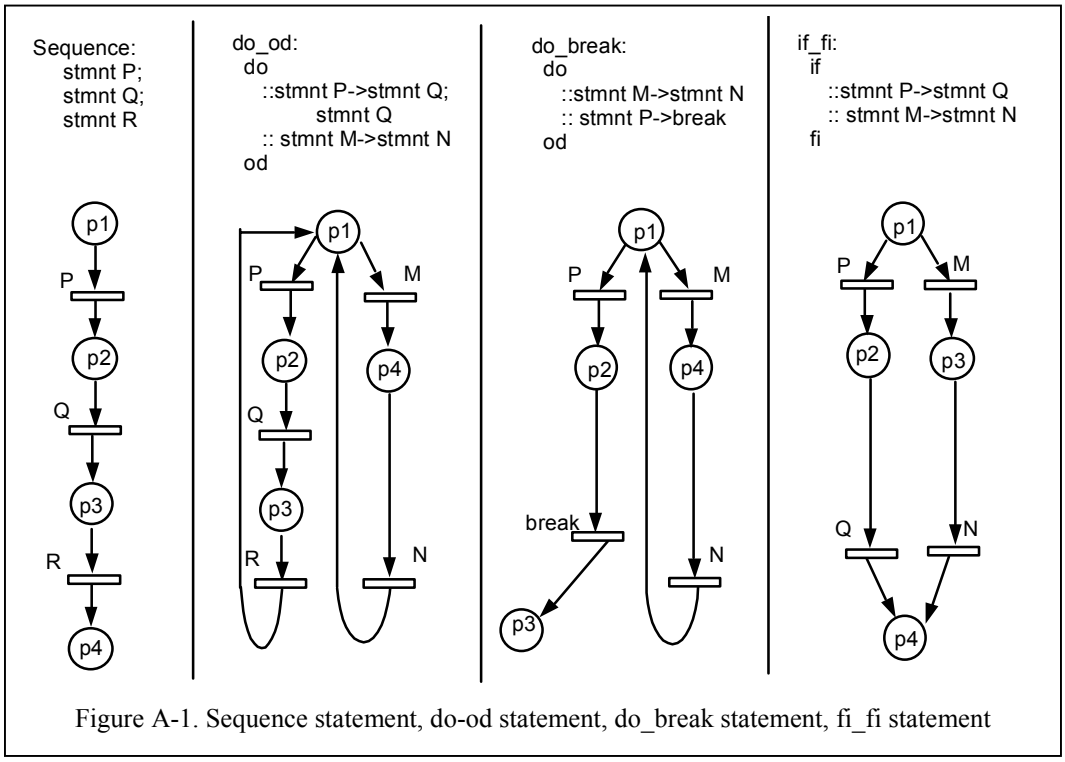

Figure A-1. Sequence statement, do-od statement, do_break statement, fi_fi statement

The do_od construct and the do_break Construct

The do_od construct and the do_break construct have conflicting transitions. The resulting SPN models are represented either with timed transitions (i.e., a rate parameter is assigned) or immediate transitions (i.e., a probability parameter is assigned). In the PROMELA model there are various

reachable states (depending on the content of statement P, Q, R, M, N), while in the PN model, there are only 4 states for both constructs. Usually, the state space in PROMELA models is much more than that in SPN models.

The if_fi Construct and the Condition Construct

The if_fi construct has same SPN representation as the condition construct. IN fact, the if_fi construct can replace the condition construct. The if_fi construct has conflict transition. The resulting SPN models are represented either with timed transition or immediate transition. During this translation PCX tool allows users to assign rates to each timed transitions, otherwise PCX tool will assign default value to them. The state space in PROMELA model is same as that in SPN model.

The goto_label Construct and the else Construct

The goto and label construct and the label construct have same situation as do_od construct and do_break construct.

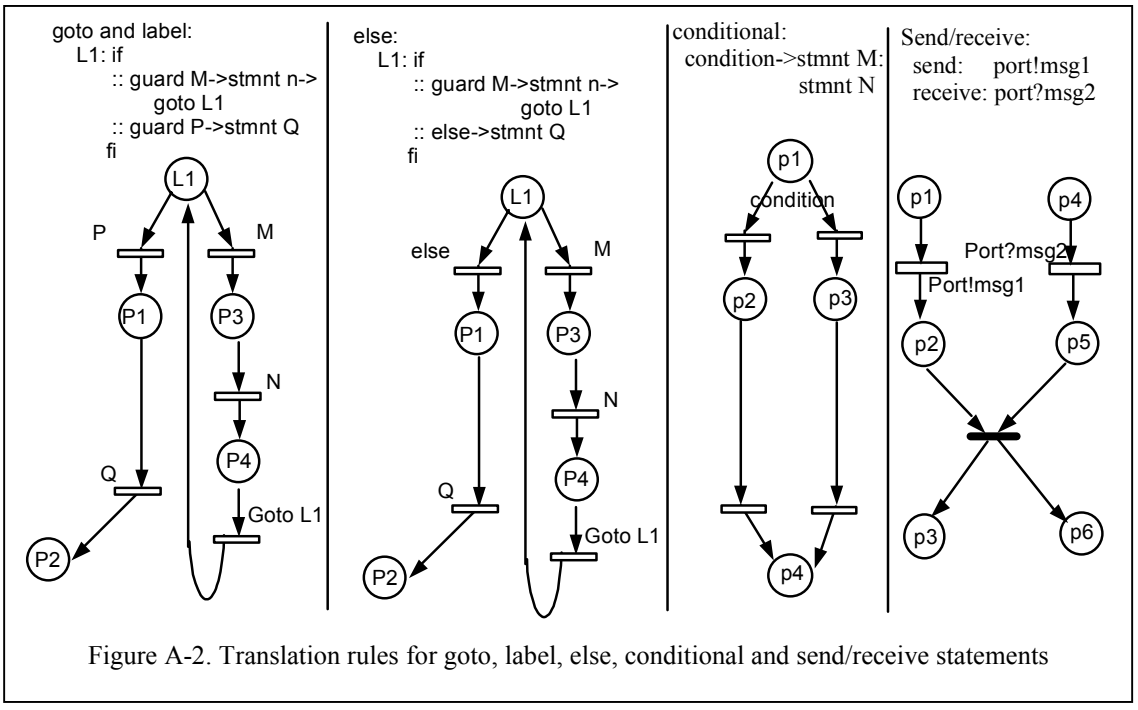

Figure A-2. Translation rules for goto, label, else, conditional and send/receive statements

Message Channel

The message construct in PROMELA transfers messages through the channels. This type of communication is modeled in a SPN by firing a transition for both of the sender and the receiver to represent the sending and receiving activity respectively. The resulting SPN models are represented either with timed transition or immediate transition. The state space in PROMELA model is same as that in SPN model.