Towards an efficient structural analysis of colored Petri nets

Mohamed Bouali, Pavol Barger, Walter Schön

International Conference
on Industrial Engineering and Systems Management
IESM 2009
May 13-15
MONTRÉAL - CANADA

Towards an Efficient Structural Analysis of Colored Petri Nets

Mohamed BOUALI, Pavol BARGER, Walter SCHON

Heudiasyc Laboratory, CNRS UMR 6599
Université de Technologie de Compiègne, France

Abstract

This paper presents the *Backward reachability* which is a structural analysis method applicable to Colored Petri Nets (CPN). This method avoids the fastidious computation of simulation and the combinatorial explosion of reachable state space. The *backward reachability* provides the information about different ways of reaching a particular CPN marking. This evaluation considers the evolution of markings participating in each sequence leading to the desired state. The analysis is performed on an *inverse CPN* which is obtained by transforming original CPN according to given inversion rules. The work develops the rules allowing the definition of inverse CPN and presents some transformations that contribute to the analysis speed up. The main advantage of this method is that it allows to determine the sequence leading from any initial state to a given final marking without investigating all possible sequences.

Key words: Petri Nets, Backward Reachability, Model Analysis

1 Introduction

Modern systems (hardware/software) integrate an ever increasing amount of components. Their goal is to enlarge the range of proposed functions and thus make the product more attractive to the consumer. The second objective is to be able to propose new product versions by updating the fewest system part. When applied to critical domain, such as car driver protection or airplane collision warning, the system has to respect required dependability criteria. For this reason formal methods appear as early as the initial system design specification. It is in this context that the presented work applies a formal method approach to system modelling and analysis.

This study is interested on Petri Nets (PN) [13], and especially, Colored Petri Nets(CPN) [7]. They are considered as a powerful and recognized modelling tool. They are endowed with a big expressiveness and allow to represent the two aspects of a system: static thanks to the PN structure and dynamic thanks to the token evolution [1]. The PN analysis can be done in several manners: exhaustive reachable state space enumeration, simulation, structural analysis, etc. These methods allow to study request/action effects on the model behavior. Usually, the performed analysis is the forward one. That means, by knowing an initial state, possible final states are calculated. This is particularly adapted to the studies of performances and the quality of service (QoS) of systems. But, it is not adapted to find sources of some particular final states which can represent a system erroneous state. In such analysis, all possible initial configurations must

Email addresses: mohamed.bouali@hds.utc.fr (Mohamed BOUALI), pavol.barger@hds.utc.fr (Pavol BARGER), walter.schon@hds.utc.fr (Walter SCHON).

be studied to find those leading to the considered failure state. This is the reason for another approach proposal for this study.

This paper proposes an approach based on the backward reachability [8,3,12]. This approach supplies reachability results without construction of the reachability graph nor forward simulation. The objective is to make a final-state driven analysis. This method focuses on the final state and calculates the initial states set from which the final state is reachable. To do this, the original CPN model is transformed, using CPN inverting rules, to express the inverse CPN on which is performed the backward reachability analysis. Thus, this approach allows a fast and efficient model analysis for a certain type of queries.

This paper is organized as follows : The section 2 gives definitions of Petri Nets and Colored Petri Nets. The section 3 presents the formal aspects of CPN, synthesizes the direct model analysis and discusses its limits. It is followed, in the section 4, by the details of our approach and, in section 6, a presentation of a sample application example. The paper ends by the conclusion and an outline of future perspectives.

2 Definitions

2.1 Petri Nets

A Petri Net [13], called also Place/Transition Net, is a directed bipartite graph defined by the 4-tuple $(P, T, Pre, Post)$, where:

- P is a finite set of places,
- T is a finite set of transitions,
- $P \cap T = \emptyset$,
- Pre is the backward incidence application,
- $Post$ is the forward incidence application.

2.2 Colored Petri Nets

The notation of Colored Petri Net (CPN) [7] introduces the notion of token types, namely tokens are differentiated by colors, which may be arbitrary data values. Each place has an associated type determining the kind of data that the place may contain. A non-hierarchical CPN is defined by the 9-tuple $(\Sigma, P, T, A, N, C, G, E, I)$ where :

- Σ is a finite set on non-empty types,
- P is a finite set of *places*,
- T is a finite set of *transitions*,
- A is a finite set of *arcs* such that: $P \cap T = P \cap A = T \cap A = \emptyset$,
- N is a node function. It is defined from A into $P \times T \cup T \times P$,
- C is a *color* function. It is defined from P into Σ,
- G is a *guard* function. It is defined from T into expressions such that:

$$\forall t \in T : [Type(G(t)) = B \wedge Type(Var(G(t))) \subseteq \Sigma],$$

- E is an *arc expression* function. It is defined from A into expressions such that:

$$\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$$

  Where $p(a)$ is the place of $N(a)$,
- I is an *initialization* function. It is defined from P into closed expressions such that:

$$\forall p \in P : [Type(I(p)) = C(p)_{MS}]$$

For practical reasons, we write E in a split form Pre and $Post$ (as used in [6]) such that Pre (resp. $Post$) is the backward (resp. forward) incidence application. It is defined as E where $p(a)$ is the place of a part of $N(a)$ defined from A to $P \times T$ (resp. $T \times P$).

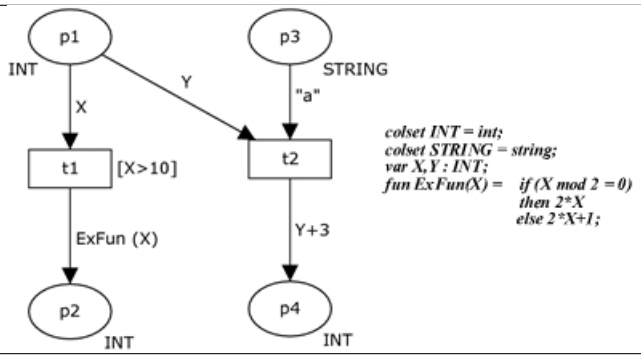

Fig. 1. An example of CPN

Fig.1 shows an example of CPN designed with the CPNtools [1] . It contains a set of four places $\{p1, p2, p3, p4\}$ and a set of two transitions $\{t1, t2\}$. Each place is declared from some defined type (called *colset*). Two types are defined: *INT* and *STRING*. The transition $t1$ is associated to the guard $[X > 10]$. That means, only tokens which values satisfy this guard can be fired through $t1$. We note also the arc expressions. X and Y are declared INT variables, $ExFun$ is a function transforming the X value and "a" is a $STRING$ constant.

3 CPN model analysis

Formal methods are based on mathematical expression of the model proprieties. They allow the creation of specifications and complete and non ambiguous models. CPN can be a starting point for a future formal analysis. The construction of a reachability graph approaches formal methods for the PN behavioral analysis. Some properties, such as liveness and fairness, can be verified. This approach is accepted within an industrial context [2] as well as in fundamental research domains [11]. Another approach suitable for flexible and automatic analysis in connection with formal methods is *model checking* [4] applied to PN. It allows to verify if the PN behavior is in accordance with a specification given by a logic. Several types of logics exist e.g. LTL (*Linear-time Temporal Logic*) [10] and CTL (*Computation Tree Logic*) [5].

State enumeration based methods are also of interest in the PN and the CPN analysis. They allow to cover all the reachable state space but suffer from the combinatorial explosion. To avoid this problem, we propose the use of the structural analysis using directly the model itself. In this case, several types of transformations can be defined. The trivial transformation is the CPN unfolding but this operation increases the model size. [6] defines rules and mechanisms to reduce CPN. This reduction is proved to preserve certain CPN properties such as liveness. It uses several mechanisms for example the post-agglomeration and the pre-agglomeration. Their purpose is to diminish the model size in order to make the formal analysis more efficient and faster when applied on a *reduced* CPN which has the same properties as the original CPN.

Another use of the structural analysis is presented in [8] which is interested in the reachability between two markings studied under two dual manners: the forward reachability and the backward reachability. The first case consists in building state successors one by one, starting with the initial marking M_0 and ending with the final marking M_f. Thus, considering M_0 as the present state, M_f is considered as the future state. In the backward reachability, M_f constitutes the present state and M_0 is regarded as the past state. The general idea is to build, from the present state, the predecessor states until the reach of the past state which is logically the source of the present marking. This study shows that the forward reachability (from M_0 towards M_f) is not necessarily identical to the backward reachability (from M_f backwards to M_0). That is in both cases, the markings and the sequences investigated are not necessarily identical. Generally speaking, the forward reachability allows to underline the marking paths which lead to the final marking and those that lead to other markings than the final marking. Likewise, the backward reachability identifies the paths which lead from the initial marking to the final marking, and also those who lead to the final marking and do not start with the initial marking. It is to note that the intersection of path sets produced by these two approaches is not empty.

[1] http://wiki.daimi.au.dk/cpntools/cpntools.wiki

In order to perform the backward reachability analysis, the inverse Petri Net (inverse PN) has to be constructed as follows:
Considering a PN R such as $R = (P, T, Pre, Post)$, the inverse PN (noted R^{-1}) is defined by $R^{-1} = (P', T', Pre', Post')$ [8] where: $P' = P$ and $T' = T$.
The applications Pre' and $Post'$ are defined as :

$$\forall p \in P, \forall t \in T \begin{cases} Pre'(p,t) = Post(p,t) \\ Post'(p,t) = Pre(p,t) \end{cases}$$

This definition cannot be directly applied to CPN. The definition of R^{-1} (in particular Pre' and $Post'$) is not sufficient as soon as arc expressions differ from the identity function. In addition to this, the backward reachability uses some mechanisms like the *"marking enhancement"* [8]. These mechanisms are efficient in the case of PN but they are incompatible with CPN because of the token value evolution due to arcs valuation. Finally, both PNs R and R^{-1} keep the same structures even though some reductions or transformation might be able to make analysis more efficient.

4 Approach presentation

As defined in PNs, backward reachability in CPNs is a dual concept of forward reachability. That is, if a marking M_f is reachable from M_0, we say that M_0 is backward reachable from M_f. By backward reachability, we mean that M_0 is a cause or source of M_f. For example, In Fig.1 consider a marking $M_f = \langle 8 \rangle.p4$ [2] . M_f is reachable from $M_0 = \langle 5 \rangle.p1 + \langle "a" \rangle p3$ and we say that M_0 is backward reachable from M_f.

Our contribution concerns firstly the generalization of the backward reachability from the PN class to the CPN class and secondly the development of adapted transformation rules. The generalization implies a definition of an inverse CPN which is obtained by performing the proposed transformations on the original CPN.

4.1 CPN transformation for the definition of an inverse CPN

Let $R = (\Sigma, P, T, A, N, C, G, E, I)$ be a CPN and let $R' = (\Sigma', P', T', A', N', C', G', E', I')$ be the inverse CPN of R. To define R', we perform some transformations on R. These transformations are directly dependent on the CPN structure. Consequently, we have to define a transformation for each structure case studied. Nevertheless, the most common transformations in representative cases are presented in this section. Note that, in the remaining paper, E is written in its split form $(Pre, Post)$. So, we write also $E' = (Pre', Post')$.

4.1.1 Basic transformations

Fig.2.a shows a trivial case of a CPN inversion. Arcs (input and output) are marked with constants a, b. In this case, it is enough to generalize the definition of the inverse PN to CPN obtaining :

$$\begin{cases} Pre'(p2,t) = Post(p2,t) \\ Post'(p1,t) = Pre(p1,t) \end{cases}$$

Note that, in the original CPN, $M_0 = \langle a \rangle.p1$ gives $M_1 = \langle b \rangle.p2$ by firing t. In the inverse CPN $M_1 = \langle b \rangle.p2$ gives $M_0 = \langle a \rangle.p1$ by firing t. Thus, we performed the backward reachability using the inverse CPN.

Fig.2.b shows the case where the input arc is marked with a variable x and the output arc is marked by a function $f(x)$. Generalized application of the rule shown before (inverting arcs directions) can lead to the construction of an incorrect net. In this example, the input arc would be marked with a function while the output arc would be marked with the variable of this function. That leads to the impossibility

[2] The formula $M_i = \langle V1 \rangle.p1 + \langle V2 \rangle.p2$ means the marking M_i contains token in $p1$ whose value is $V1$ and token in $p2$ whose value is $V2$ (notation taken from [9])

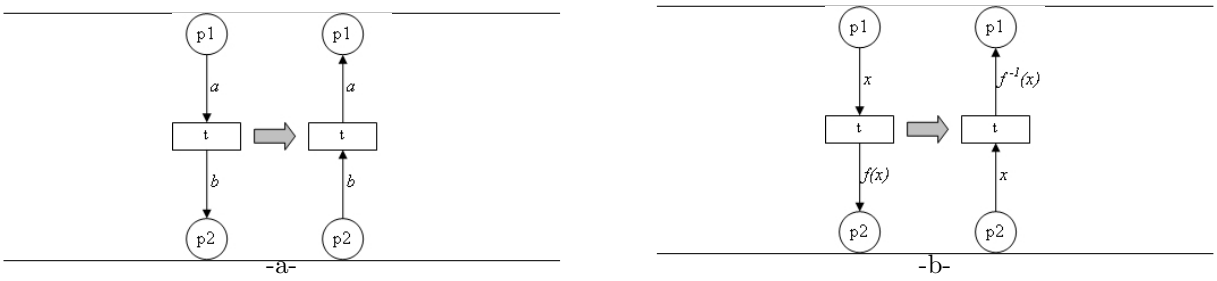

Fig. 2. Basic transformations for CPN inversion

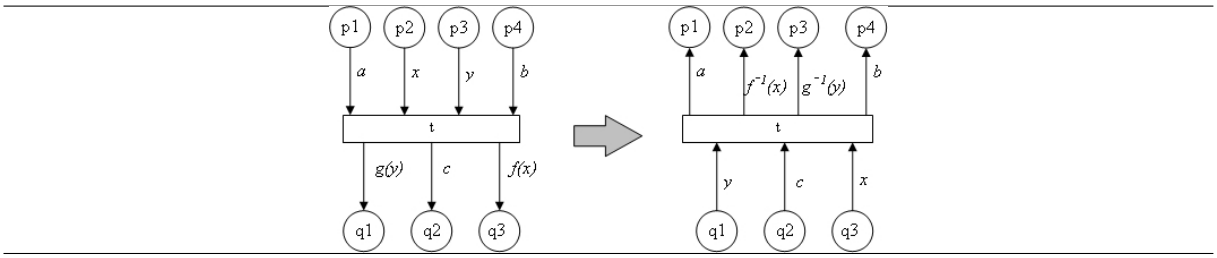

Fig. 3. Mixed transformations for CPN inversion

for the function evaluation. This is why we suggest marking the input arc by the variable and the output arc by the function f^{-1}. This assumes the necessity to know whether function f is reversible. If yes, it is necessary to define its inverse noted f^{-1}. This transformation gives :

$$\begin{cases} Pre'(p2,t) = Pre^{-1}(p1,t) \\ Post'(p1,t) = Post^{-1}(p2,t) \end{cases}$$

$Pre^{-1}(p1,t)$ denotes transformation of the arc $Pre(p1,t)$ which is marked with a new variable defined on f^{-1} domain. $Post^{-1}(p2,t)$ denotes transformation of the arc $Post(p2,t)$ which is marked by f^{-1} (the opposite reverse of $Post(p2,t)$ marking function).

4.1.2 Mixed transformations

Fig.3 shows a mixed case where some input arcs are marked with variables $\{x, y\}$ and other arcs by constants $\{a, b\}$. Output arcs are marked with constants $\{c\}$ and reversible functions $\{f, g\}$. This CPN inversion is a mix (generalization) of basic transformations. For arcs marked with a constant, it is sufficient to generalize the rule applied on PNs (changing arcs direction). For the remaining arcs, we have to apply the rule illustrated in Fig.2.b. So, we have to associate each variable to its function in order to respect origine places of arcs marked by variables and sink places which receive resulting tokens. The following constraint has to be verified in this case: each variable is used by one and only one function. If not true, see the *Parametric transformation* subsection below. The algorithm describing the mixed transformation can be written as follow :

- $Pre'(qj,t) = Post(qj,t)$, $j = 1 \dots m$ if $Post(qj,t)$ is marked with a constant,
- $Post'(pi,t) = Pre(pi,t)$, $i = 1 \dots n$ if $Pre(pi,t)$ is marked with a constant,
- $Pre'(qj,t) = Pre^{-1}(pi,t)$, $i = 1 \dots n$, $j = 1 \dots m$ if $Pre(pi,t)$ is marked with a variable and this variable is associated to the function $Post(qj,t)$,
- $Post'(pi,t) = Post^{-1}(qj,t)$,$i = 1 \dots n$, $j = 1 \dots m$ if $Post(qj,t)$ is marked with a function and this function is associated to the variable $Pre(pi,t)$.

4.1.3 Parametric transformations

Some CPN structures can't be reversed to get deterministic markings in the backward reachability. The reason is that the inversion process could be assimilated to mathematic operations whose solutions may be intervals. The CPN inversion, in these cases, is *parametric*. That means, some additional information,

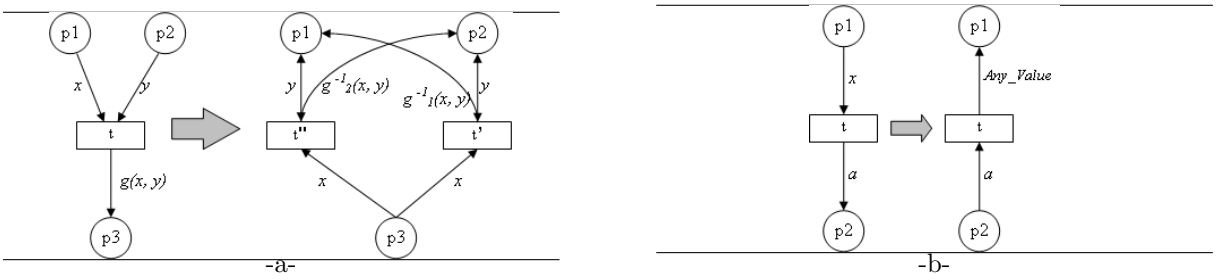

Fig. 4. Parametric transformations for CPN inversion

like color sets, are needed. Two examples follow : the first treats solution of multivariable equations, the second treats input variables which are not associated with output functions.

Fig.4-a- shows a case where output arc is marked by a multivariable function. In this case, the CPN is not directly reversible. The values of variables $\{x, y\}$ can't be deduced by knowing only the value of $g(x, y)$. The *partial (parametric)* solution proposed consists to find values of one variable knowing other variable values and the function result. As illustrated in Fig.4-a-, firing t' (in inverse CPN) requires tokens in $p2$ and $p3$. Token in $p2$ is considered as the known variable. Token in $p3$ is considered as the resulted token by firing t. Firing t' produces two tokens. The first token value, in $p1$, is calculated through arc expression $g_1^{-1}(x, y)$ which is the first *partial inverse* of $g(x, y)$.

Fig.4-b- shows a case of variable which is not associated with a function. We note that input arc expression is a variable and output arc (or arcs) expression is constant. This CPN inversion is similar to the case of Fig.2.a, except the arc $Post'(p1, t)$ which is marked by a *parametric* expression noted $'Any\_Value'$. It means that this arc expression can take all values out of t in the place $p1$ color set.

4.1.4 Parallelism transformations

The term *'parallelism'* means existence of a shared variable. Fig.5 shows the case where the same variable is used by more than one function (two functions in this case). To inverse this CPN, we have to calculate the inverse of only one function using the shared variable (that supposes existence of, at least, one reversible function). Let f be a reversible function. In the original CPN (shown on the left in Fig.5), transition t firing produces two tokens (in $p2$ and $p3$). Each token value results from a function applied to the shared variable : function f to the token in $p2$ and function h to the token in $p3$. The inverse CPN must produce a token in $p1$ by firing t whose preconditions are $p2$ and $p3$. But it is not enough to have tokens in $p2$ and $p3$ to fire t because token values (in $p2$ and $p3$) must have coherent values towards applied functions f and h. For this reason, we define a *guard* associated to transition t. It checks that the value of $f^{-1}(x)$ applied to h gives as the result the value of y (token initially in $p3$). If the guard value is *True*, t will be fired, then the initial marking of $p1$ will be found. If the guard value is *False*, t will not be fired.

To illustrate this transformation, let extend the example of Fig.5 where $f(x) = x + 4$, $h(x) = (x > 10)$ and initial marking $M_0 = \langle 5 \rangle.p1$ The evolution of this marking is :

$$\begin{aligned} M_0 &= \langle 5 \rangle.p1 \\ &\quad t \downarrow \\ M_f &= \langle 9 \rangle.p2 + \langle False \rangle.p3 \end{aligned}$$

The first test consists to apply backward reachability to M_f on the inverse CPN (Fig.5). The expected result is to determine the initial marking M_0.

$$\begin{aligned} M_f &= \langle 9 \rangle.p2 + \langle False \rangle.p3 \\ &\quad t \downarrow \\ M_0 &= \langle 5 \rangle.p1 \end{aligned}$$

Note that the guard gives $'True'$. It is obtained by applying the formula $[y == h(f^{-1}(x))]$ which is $['False' == (5 > 10)]$.

The second test consists to apply backward reachability to an unreachable final marking M'_f on the inverse CPN of the figure. The expected result is to find an empty set of initial markings M_0.

$$M'_f = \langle 9 \rangle.p2 + \langle True \rangle.p3$$
$$t \downarrow$$
$$\{\}$$

$\{\}$ stands for impossible firing. Note that the guard gives $'False'$. It is obtained by applying the formula $[y == h(f^{-1}(x))]$ which is $['True' == (5 > 10)]$.

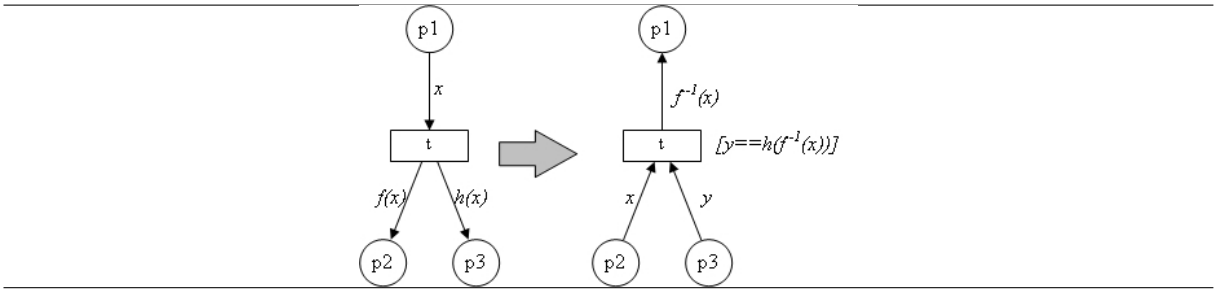

Fig. 5. Parallelism transformation for CPN inversion

The algorithm describing the parallelism transformation can be written as follow:
Let R be a CPN containing a variable x $(Pre(p,t) = x)$ which is shared by n functions $f_1, f_2, \ldots, f_n$ $(Post(qi,t) = f_i(x), i = 1 \ldots n)$ with f_1 inversible. The inverse CPN is defined by :

- $Pre'(q1,t) = Pre^{-1}(p,t)$,
- $Pre'(qi,t) = Id_i, i = 2 \ldots n$,
- $Post'(p,t) = Post^{-1}(q1,t)$,
- $Guard(t) = \bigwedge_n^{i=2}[Pre'(qi,t) == f_i(f_1^{-1}(Pre'(q1,t)))]$.

4.1.5 Loop transformation

The models aimed with this study show a recurrent presence of small loops (usually representing timers). Their minimal form is a loop between a place and a transition representing a very simple timer or sampler. To avoid a repetition of iterations, we propose the use of sequences, and especially, arithmetic and geometric sequences allowing to determine in only one operation the final value after a given number of iterations or, by duality, to calculate the number of necessary iterations to reach a value without having to simulate the loop.

The concerned loops have a progression which can be compared to a sequence. For example, in Fig.6-a-, we can put $\langle U_0 \rangle.p$ as initial marking. The successive firing of t produces tokens which values are $U_1 = f(U_0)$, then $U_2 = f(U_1) = f(f(U_0)), \ldots, U_n = f(U_{n-1}) = \underbrace{f(\ldots(f}_{n\ times}(U_0)\ldots)$. In some cases according to the form of the function f, values $U_i, i = 1, \ldots, n$ can be considered as terms of sequences and can be deduced easily.

In the case of an elementary (Fig.6), the CPN is transformed by duplicating the place constituting the loop and the input and output arcs. In the example of Fig.6, the result is shown in Fig.6.b. The place p is duplicated in p, p' and p". The function $f(x)$ is transformed in $f^*(x, U_0)$. The function f^* calculates the number of necessary iterations (steps) to reach the initial value of the token in p.

If, for example, $f(x)$ is written as $f(x) = x + b$ (x is a numeric variable, b is numeric constant) then $f^*(x, U_0) = \frac{x - U_0}{b}$ is the value of p". U_0 is the initial value in p' and it constitutes the input value of the loop. Let note that to have the literal formula of f^*, it is enough to consider f, which characterizes the loop, like an arithmetic progression with common difference b and initial value U_0.

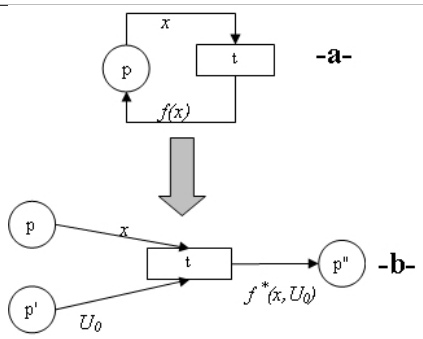

Fig. 6. Elementary loop before and after transformation

The algorithm describing the loop transformation can be written as follows :
Let $(\Sigma, P, T, A, N, C, G, E, I)$ be a CPN and $(p, t) \in P \times T$ an elementary loop elements which behavior is comparable to a sequence. In the inverse CPN $(\Sigma', P', T', A', N', C', G', E', I')$

- Duplicate p in p, p' and p",
- $Pre'(p, t) = Pre(p, t)$,
- $Pre'(p', t) = U_0$ such that U_0 is a variable which takes its values in the domain of p,
- $Post(p", t) = f^*(x, U_0)$ such that $Post(p, t) = f(x)$.

5 Supplementary mechanisms

The rules presented above allow the inversion of the CPN structure. In order analyse the inversed dynamic behavior two complementary principles are state here.

5.1 Marking Enhancement

The marking enhancement is a mechanism allowing to complete information about a model by adding assumptions about the modelled system state (additional tokens). It is done when constructing successor states. Sometimes, the immediate successor of a given state (which is equivalent to fire a transition) can only be constructed with help of added tokens with appropriate values in certain places. The marking enhancement interpretation consists to assume that the system is in some given state allowing it to operate. This mechanism has to be performed if only a part of the token distribution is known.

5.2 Potentially Valid Transition

In a marked CPN, a transition is potentially valid is it has, at least, 1.) one precondition place marked by a token whose value is compatible with the transition firing (i.e. token compatible with arc expression and guard) ; and 2.) one precondition place which is not marked by a token whose value is compatible with the transition firing.

6 Case study

This section shows a practical using of the backward reachability (through an inverse CPN) presented in previous paragraphs. The application example is inspired from *"Simple Protocol"* [3]. It describes a communication protocol with identified acknowledgement.

The basic question studied here is whether the receiver could possibly receive a hypothetical sequence of frames.

The CPN model of the protocol is shown in Fig.7. It consists of three parts. The *Sender* part has two transitions which can *Send Packets* and *Receive Acknowledgments*. The *Network* part has two transitions which can *Transmit Packets* and *Transmit Acknowledgments*. Finally, the *Receiver* part is modelled by a

[3] http://wiki.daimi.au.dk

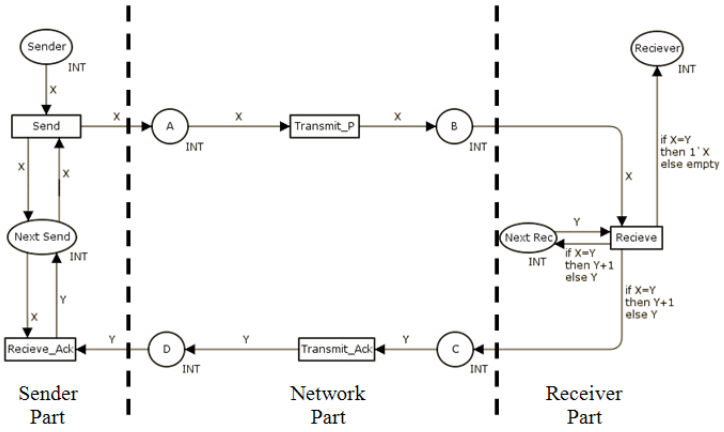

Fig. 7. CPN model of the communication protocol

single transition which can *Receive Packets* and *send acknowledgments*. The interface between the *Sender* and the *Network* consists of places A and D, while the interface between the *Network* and the *Receiver* consists of places B and C. The packets to be sent are placed in place *Send*. Each token in this place contains a packet represented by its sequence number. The place *Next_Send* contains the number of the next packet to be sent and is updated each time when the correct acknowledgment is received. All correctly received messages are stored in a buffer represented as place *Receiver*. Place *Next_Rec* contains the number of the next packet to be received and is updated each time a packet is successfully received.

The normal working of this system would not allow doublets (meaning the correct reception of two identical frames). Thus for example the marking with 3 following packets in place *Receiver* where the first is identified as 1, two others frames equal to each other and identified as 2 represent and unreachable state. In the following this marking will be noted as $\langle 1 + 2 + 2 \rangle.Send$.

To verify the reachability of the previously given marking ($\langle 1 + 2 + 2 \rangle.Send$), the backward reachability analysis is performed using the inverse CPN, illustrated in Fig.8. Inversions of $Transmit\_P$ and $Transmit\_Ack$ are basic transformations. Those of *Receive_Ack* is a case of parametric transformation (The input variable X is not used in an output function). The goal of this transition is to delete the old value in *Next_Send* and to replace it by the token value of the place D. So, in the inverse reasoning, the goal is to put in D the token value of *Next_Send* and to put in *Next_Send* an arbitrary value. The token already contained in this place is then returned. The inversion of transition *Send* is a parallel one. This is can be seen through the input variable X which is used in two output functions (toward A and *Next_Send*). This explains the new variable Z and the guard $[X = Z]$. The inversion of *Receive* is a combination of parallel and parametric (the function are parameterized by X and Y). In the inverse problem, tokens in *Receive* are known which is equivalent to consider that possible initial values of X are known.

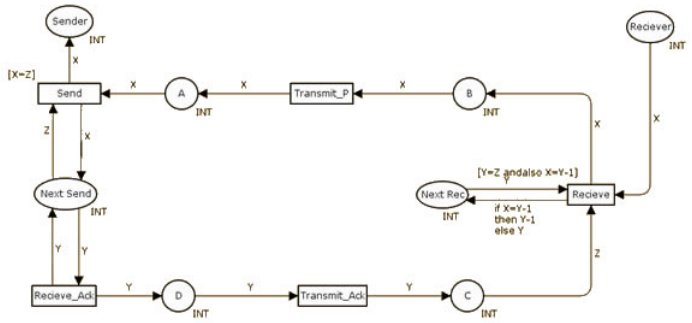

Fig. 8. inverse CPN of the communication protocol

In the original CPN, the sequence of transition firing is : *Send*, $Transmit\_P$, *Receive*, $Transmit\_Ack$, *Receive_Ack*. This corresponds to a partial order of transition firings. The inverse CPN analysis (endowed by the marking $\langle 1 + 2 + 2 \rangle.Send$) must respect the inversed order starting with transition *Receive* which is the unique potentially valid transition. The *receive* firing is done using marking enhancement. Two

tokens have to be added to the model : the first one into place $Next\_Rec$ and the second into place C. The two tokens are numbered 3 because this value is the only one that fulfills guard constraints. This marking enhancement corresponds to the assumption that the packet numbered 2 was correctly received. The result is a token of value 2 in places $Next\_Rec$ and B after firing $Transmit\_P$ in A. The firing of $Send$ requiers a token valued 2 in $Next\_Send$ which assumes a correct packet transmission. The marking evolution (without any enhancement) processes the packet numbered 1 thanks to all transitions firings (one time for each one). This leads to a deadlock situation with a packet numbered 2 in place $Receiver$.

Thus the conclusion is reached and expresses that there is no coherent initial marking possible to obtain the $\langle 1 + 2 + 2 \rangle.Send$ marking in place $Receiver$. Upon detection of such marking, an unexpected fault has to be declared.

7 Conclusion

This paper presents an extension of the backward reachability to some categories of CPN. As the backward reachability was not originally defined on the CPN the corresponding rules for are also introduced. The rules of extension, such as the definition of the colored incidence applications for the inverse CPN, allow to keep the properties and the advantages of the backward reachability and apply them to the CPN class.

The main application field for the backward reachability is the model diagnostics during the conception. The system initial state is usually well determined and so is the final state which in this case represents an undesired event. The proposed method can provide answers to the possibility of undesired event occurrence, the earliest failure time, the system evolution vector, etc.

The perspectives start with the development of an automated tool for a verification of the CPN structure in order to be acceptable for the backward reachability analysis. The theory advances include the generalization of transformation rules to non elementary loops and to composed functions. The second path to explore is the possible of presence of non deterministic time constraints for the transition firing. The study of stochastic firing vectors is also considered as a potential subject of interest.

References

[1] F. Basile, C. Carbone, and P. Chiacchio. Simulation and analysis of discrete-event control systems based on petri nets using pnetlab. *Control Engineering Practice*, pages 241–259, 2007.

[2] I. Cibrario Bertolotti, L. Durantea, P. Maggib, R. Sistob, and A. Valenzanoa. Improving the security of industrial networks by means of formal verification. *Computer Standards and Interfaces*, pages 387–397, 2007.

[3] S. M. Cho, H. S. Hing, and S. D. Cha. Safety analysis using colored petri nets. In *Proc. Asia-Pacific Software Engineering Conference (APSEC-96), 4-7 December 1996, Seoul, Korea*, pages 176–183, 1996.

[4] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, USA, 1999.

[5] E.A. Emerson and J.Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *14th ACM Symp Theory of Computing (STOC'82), San Francisco, CA, USA*, pages 169–180, 1982.

[6] S. Haddad. A reduction theory for coloured nets. *Lecture notes in Computer science n° 424 Springer-Verlag*, pages 209–235, 1989.

[7] K. Jensen and G. Rozenberg. *High-level Petri Nets : Theory and Application*. Springer-Verlag, Germany, 1991.

[8] S. Khalfaoui. *Méthode de recherche des scénarios redoutés pour l'évaluation de la sûreté de fonctionnement des systèmes mécatroniques du monde automobile*. PhD thesis, Institut National polytechnique de Toulouse (France), Septembre 2003.

[9] Sous la direction M. Diaz. *Les réseaux de Petri : modèles fondamentaux*. Hermès Science Publications, Paris (France), 2001.

[10] L. Lamport. 'sometimes' is sometimes 'not never' : On the temporal logic of programs. *7th Annual ACM Symp. on Principles of Programming Languages (POPL 80)*, pages 174–185, August 1980.

[11] T. Latvala and M. Mäkelä. Ltl model checking for modular petri nets. *Applications and Theory of Petri Nets, 25th International Conference, ICATPN'04, Bologna, Italy*, pages 298–311, 2004.

[12] N. G. Leveson and J. L. Stolzy. Safety analysis using petri nets. *IEEE Trans. Softw. Eng.*, 13(3):386–397, 1987.

[13] Carl Adam Petri. *Communication with automata*. PhD thesis, Darmstadt Institut für Instrumentelle Mathematik, Bonn (Germany), 1962.