

Property Specification for Coloured Petri Nets*

Ella E. Roubtsova

Technology Management Department
Eindhoven University of Technology

Den Dolech 2, Postbus 513, 5600MB, Eindhoven, The Netherlands

E.Roubtsova@tue.nl

Abstract – *The Coloured Petri Nets is a powerful modelling framework which allows designers to attach values of complex types (colors) to moving objects (tokens). Using analysis techniques of the classical Petri Nets for the Coloured Petri Nets demands abstracting from colors, that can be a source of errors. In this paper we propose a logic and a tool prototype to formulate properties of a Coloured Petri net without abstracting from colors. These properties can be used both for checking correctness and for planning and controlling the simulation process. We show examples of properties for a simplified Coloured Petri Net of a distribution center.*

Keywords: Coloured Petri Nets, Temporal Logic, Properties of Coloured Petri Nets, Property Diagram, Goals and Feedback in Simulation.

1 Introduction

The Coloured Petri Nets (CPN) is a powerful modelling framework, which combines advantages of the classical Petri Nets (PN) with the expressive power of complex data types (*colors* [1]). The analysis techniques for the CPN have been inherited from the PN [1]. Those techniques usually abstract from *colors*. However, the correct abstraction from colors cannot be fulfilled until the moment when the properties of a CP-net have been formulated. The matter is that the quantitative elements of *colors* have different forms and abstracting from colors an analyst should prove that the chosen abstraction preserves the properties of the initial CP-net.

The situation with specification of properties is the following. On the one hand, without abstracting from *colors* the definitions of some properties, traditionally used for the classical Petri Nets, like safety, soundness or invariants, are not applicable to the CPN. For example, the classical definition of safety [2], which is each place contains not more than one token, does not guarantee that the system is safe from overflowing, because one token of type *list* can have an infinite number of elements. On the other hand, data types allow designers to express additional properties of their models that

have no analogies in the classical PN. Taking into account that the number of type combinations is infinite, it is impossible to give definitions of CPN properties in general. The solution is to give designers a tool to formulate properties for the CPN.

Property specification is an important part of system specification which can be used to make abstractions, to verify models of the system and to focus and organize the simulation process using those models. To specify properties of a system designers use a logic. Kindler et al [3] have investigated the state-based and event-based logics with the semantics of *the occurrence net* corresponding to a classical Petri Net. In this paper we propose a temporal logic with the reachability graph semantics for the Coloured Petri Nets. Our logic is state-based. It is a suitable language for describing both the properties of CPN that are similar to properties of the classical PN and also the additional properties which have no analogies in the classical PN. We adopt the temporal logic to extend the today's CPN tools by a part for property specification and property-oriented simulation with and without abstracting from colors.

The paper is organized as follows. Section 2 starts with an example of a CP-net, a simplified model of a distribution center. A reachability graph of a CP-net and a logic built on the top of this graph are defined in this section. Section 3 describes a property diagram and a tool prototype to specify properties. This section also discusses the role of properties in planning and controlling the simulation process. Section 4 is the conclusion.

2 A CP-net and Its Internal Model

2.1 An Example

Let us present a simplified CP-net of a distribution center. The main function of a distribution center is delivering orders requested by clients. For this purpose there is a stock containing some amount of products. The internal functions of a distribution center are to replenish the stock and to keep the information about

the debt that can not be delivered until the moment when the requested amount of products is available on the stock.

For the sake of simplicity, firstly, we assume that the stock of the distribution center contains only units of one sort of product. This simplification allows us to better illustrate the use of properties specification for the CPN. Secondly, we abstract from the replenishment policy of this stock and represent the replenishment system by transition *supply* (Figure 1). This transition increases the number of units in the stock by 200 units. At last, we abstract from a possible backordering policy [4], which organizes internal queues of orders that should be delivered.

So, the simplified stock can be modelled by one token of *color* $Q=INT$ in place *stock*. (Figure 1). The quantity of the product in the stock is represented by variable d and the value of this variable is named *on-hand stock* [4].

Arrived orders are modelled in Figure 1 by tokens in place *order*. Each order is a pair (i, c) of *color* $ORDER = product\ ID * Q$. For example, order $(\text{"A"}, 40)$ contains an order-identifier $i = \text{"A"}$ of *color* $ID = STRING$; and an order-quantity $c = 40$ of *color* $Q = INT$;

Place *delivery* represents deliveries received by customers.

An order is delivered completely by transition *deliver* only if *on-hand stock* is not empty $d > 0$ and more or equal to the order-quantity $d \geq c$ and the current debt is zero $r = 0$. The guard of this transition is

$$d > 0 \text{ andalso } d \geq c \text{ andalso } r = 0.$$

An order is delivered partly by the transition *partly* if current *on-hand stock* d is not sufficient to deliver complete order $d < c$ and the current debt is zero $r = 0$. The guard of transition *partly* is

$$d > 0 \text{ andalso } d < c \text{ andalso } r = 0.$$

The current debt of the distribution center is indicated by a token in place *debt* and represented by variable r . The debt has priority with respect to orders to be delivered. The priority is guaranteed by the expression $r = 0$ of the guards of transitions *deliver* and *partly*. Transition *rest* models delivery of the current debt if the stock has sufficient amount of the product, i.e. the transition *rest* fires if its guard is true:

$$d > 0 \text{ andalso } d \geq r \text{ andalso } r > 0.$$

The properties of the model are defined by the goals of the distribution center. For example, each order (i, c) in the model should be delivered as a finite set of deliveries $(i, c_1), (i, c_2), \dots, (i, c_n)$ such that the sum of quantities $c_1 + \dots + c_n$ is equal to the order quantity c . Formulating this property is problematic when abstracting from

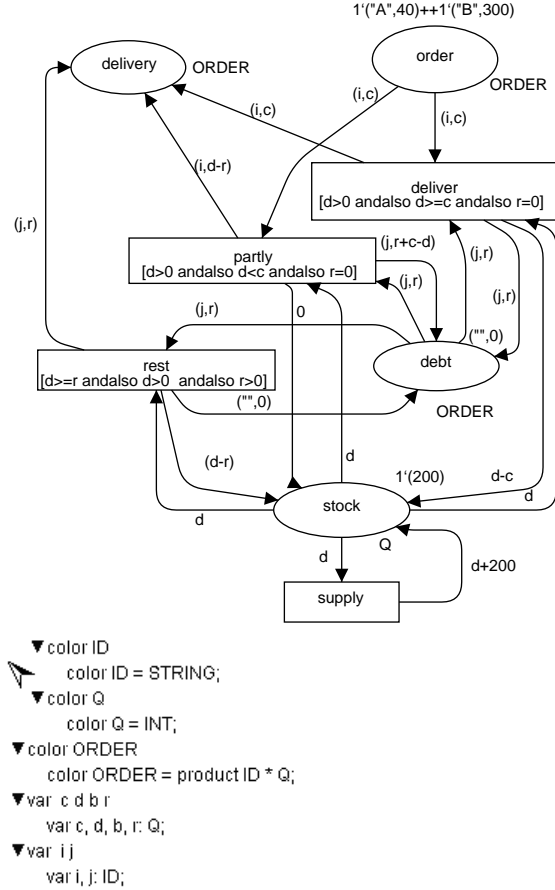


Figure 1: CP-net of a distribution center

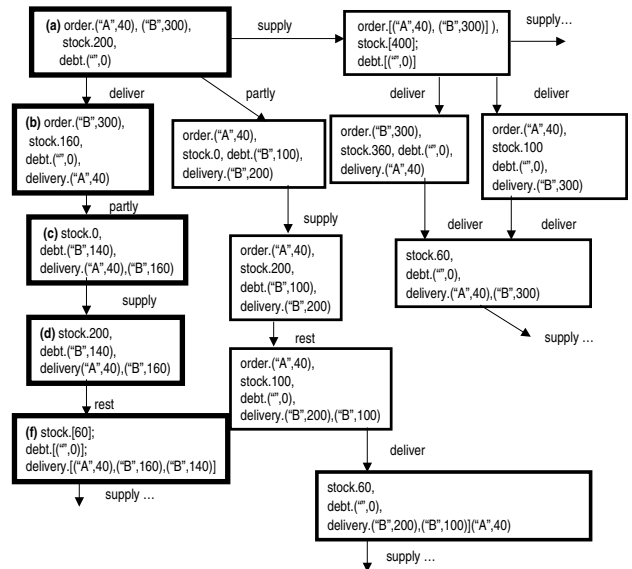


Figure 2: Reachability graph for the CP-net of a distribution center

colors. So, to formulate properties for the CPN we need an approach which is different from the approach used for the classical PN. In the next sections we define a logic which allows designers to represent properties of the CPN.

2.2 A CP-net

In order to define a logic to reason about the CPN we give the definitions of a CP-net and a reachability graph of a CP-net. We restrict ourselves by a subclass of the non-hierarchical CPN without time stamps to simplify the explanation of the logic that we introduce.

An initialized non-hierarchical CP-net without time stamps is a tuple

$$CPN = (C, B, V, P, T),$$

- C is a finite set of colors (data types) $c \in C$.
- V is a set of variables of colors $c \in C$. Colors and variables are represented as declarations (Figure 1).
- B is a bag of tokens (values) of colors $c \in C$, represented near some places. For example, place *stock* contains the token of value 1'200 (notation of the CPN-tool [1]).
- P is a finite set of places, $p_1, \dots, p_m \in P$, depicted by ellipses (Figure 1). Each place p possesses a bag $b_p \subseteq B$ of tokens of color c_p . We represent bag b_p of place p using the dot-notation $p.b_p$ and name it *place-marking*. By default bag b_p is empty.
- T is a finite set of transitions depicted by boxes (Figure 1).

Each transition is a tuple $t = (It, gt, Ot)$ (Figure 3), where

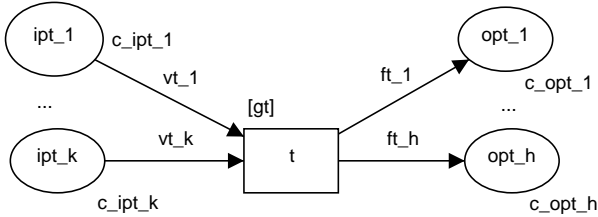


Figure 3: A transition of a CPN

- It is a finite set of input arcs. An input arc $(ipt_k, vt_k, t) \in It$ is directed from place ipt_k to transition t (Figure 3) ($k = 0, 1, \dots, K$). An arc is a triple of place $ipt_k \in P$, input arc expression $vt_k : c_{ipt_k}$ of the color of the corresponding input place and transition t .
- gt is a guard of transition t . Each guard is a boolean function:

$$gt : c_{ipt_1} \times \dots \times c_{ipt_k} \rightarrow \{true, false\}.$$

By default each guard has value *true*.

- Ot is a finite set of output arcs (Figure 3). An output arc $(t, ft_h, opt_h) \in Ot$ is a triple containing transition t , place $opt_h \in P$, $h = 0, 1, \dots, H$ and arc description ft_h , which is a function

$$ft_h : c_{ipt_1} \times \dots \times c_{ipt_k} \rightarrow c_{opt_h}.$$

Behaviour of a CP-net.

Transition t of a CP-net is enabled if places of all its input arcs contain tokens to give values to input expressions of t . Each enabled transition can fire.

When transition t fires:

- For each input arc (ipt_k, vt_k, t) its input expression vt_k is evaluated by a token from the place ipt_k .
- For each output arc (t, ft_h, opt_h) , its output function ft_h is calculated using values vt_k and the result of the output function is added as a token into the place opt_h of the output arc.

2.3 Reachability Graph

All possible sequences of firings in a CP-net can be represented by the reachability graph of the CP-net.

A reachability graph of a CP-net is a graph

$$R = (M, E), \quad M - \text{a set of nodes and } E - \text{a set of arcs.}$$

A node $m \in M$ is a tuple of place markings: $p_1.b_{p_1}, \dots, p_n.b_{p_n}$. The places with empty bags of tokens are omitted from the nodes of the reachability graph. For example, the initial node m_0 of the reachability graph in Figure 2 is represented by the following tuple of place markings:

$$m_0 = \text{order.}(\text{"A"}, 40), (\text{"B"}, 300), \\ \text{stock.}200, \text{debt.}(\text{""}, 0).$$

An arc $e = (m_1, m_2, t) \in E$, $m_1, m_2 \in M$ is a pair of nodes labelled by a transition t , such that m_2 is reachable from m_1 as a result of firing of this transition t . For example, arc $(m_0, m_1, \text{deliver})$ of the reachability graph (Figure 2) shows that from the initial node m_0 after firing transition *deliver* we reach node

$$m_1 = \text{order.}(\text{"B"}, 300), \text{stock.}160, \\ \text{debt.}(\text{""}, 0), \text{delivery.}(\text{"A"}, 40).$$

The reachability graph is a suitable model for CPN property specification, because it contains the information about values of tokens and the information about behaviour of the model.

2.4 A Logic Built on the Top of a Reachability Graph

Defining a logic of some specific field it is very important to choose the right atomic propositions. Formulating properties, CPN designers usually observe a subset of important places and compose a marking from those

places as an initial point for a property. That is why, we define an atomic proposition ϕ :

$$X.b_X [p]$$

which describes a place marking: *there exists place X which contains bag b_X such that predicate $p = \text{true}$.* We omit the quantifier $\exists : \exists X.b_X [p]$ and symbol \in : $b_X[p] \in X$ in atomic propositions to simplify them. The predicate p can contain the information about the colors of tokens from bag b_X and additional model information. If p contains only the color information corresponding to X , then p can be omitted, because place X in the syntactically correct CP-net contains only tokens of its own color. The negation of the atomic proposition $\neg X.b_X$ expresses the fact that place X does not contain some specific bag. We define function $m(\phi)$ which derives the marking corresponding to proposition ϕ .

If the color of a place of a CP-net is complex, then the variables of *sub-colors* are used to represent its tokens. For example, the atomic proposition:

$$\begin{aligned} & \text{delivery.}(i, c_1), \dots, (i, c_n) \\ (i : \text{STRING} \wedge c_i : \text{INT} \wedge c_1 + \dots + c_n = 50) \end{aligned}$$

means that place *delivery* contains a set of deliveries with the same identifier i and the sum of quantities of the deliveries is equal to 50 units.

We transform a reachability graph of a CP-net into a Kripke structure [5]: $\mathcal{M} = (M, E, \mu)$, where (M, E) is a reachability graph and $\mu : M \rightarrow 2^{TP}$ is a function which assigns true values of propositions to each node of the reachability graph. Let us recall that a node of a reachability graph is a marking of the corresponding CP-net. Function μ sets that a proposition ψ is true in making m corresponding to node m of a reachability graph if and only if the place marking described by the proposition is a sub-bag of marking m : $m(\psi) \subseteq m$. For example, atomic predicate *order.("A", 40)* describes a marking which is a sub-bag of the initial marking

$$\begin{aligned} m_0 = & \text{order.}(\text{"A"}, 40), (\text{"B"}, 300), \\ & \text{stock.}200, \text{debt.}(\text{""}, 0) \end{aligned}$$

of the reachability graph shown in Figure 2, because $\text{order.}(\text{"A"}, 40) \subseteq m_0$.

Let TP (abbreviated from token predicates) be a set of atomic propositions $\phi \in TP$. The formulas ψ, ψ_1, ψ_2 of the logic are inductively defined as follows

$$\psi ::= \phi \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \psi_1 AU \psi_2 \mid \psi_1 EU \psi_2.$$

The satisfaction relation for formulas in markings $(\mathcal{M}, m) \models \psi$ is derived inductively:

1. $m \models \phi$ iff $\phi \in \mu(m)$;
2. $m \models \neg\psi$ iff $m \not\models \psi$;

3. $m \models \psi_1 \vee \psi_2$ iff $m \models \psi_1$ or $m \models \psi_2$;
4. $m \models \psi_1 \wedge \psi_2$ iff $m \models \psi_1$ and $m \models \psi_2$;
5. $m \models \psi_1 AU \psi_2$ if for every path m_0, m_1, \dots with $m = m_0$ for some $i \geq 0 : m_i \models \psi_2$ and for some $0 \leq j < i : m_j \models \psi_1$.
6. $m \models \psi_1 EU \psi_2$ if for some path m_0, m_1, \dots with $m = m_0$ for some $i \geq 0 : m_i \models \psi_2$ and for some $0 \leq j < i : m_j \models \psi_1$.

The interpretation of the satisfaction relation in our logic has the reachability graph semantics:

1. predicate ϕ is satisfied in all nodes (markings) m of the reachability graph where the marking $m(\psi)$ described by ψ is a subbag of m : $m(\psi) \subseteq m$.
2. predicate $\neg\psi$ is satisfied in all markings m where predicate $m(\psi) \subseteq m$ is false.
3. predicate $\psi_1 \vee \psi_2$ is satisfied in all markings where ϕ_1 or ψ_2 is satisfied.
4. predicate $\psi_1 \wedge \psi_2$ is satisfied in all markings where both ϕ_1 and ψ_2 are satisfied.
5. predicate $\psi_1 AU \psi_2$ is satisfied in marking m if for every path m_0, m_1, \dots of the reachability graph starting from marking $m = m_0$ there is marking m_i such that for markings m_0, \dots, m_{i-1} predicate ψ_1 is true and for m_i predicate ψ_2 is true.
6. predicate $\psi_1 EU \psi_2$ is satisfied in marking m if for some path m_0, m_1, \dots of the reachability graph starting from marking $m = m_0$ there is marking m_i such that for markings m_0, \dots, m_{i-1} predicate ψ_1 is true and for m_i predicate ψ_2 is true.

Derivable predicate $AF\psi$ means that on all paths, there is a node, satisfying ψ . Predicate $EF\psi$ is used, if there is a path, on which there is a node, satisfying ψ . Predicate $AG\psi$ means that all nodes on all paths satisfy ψ . Predicate $EG\psi$ means that there is a path where all nodes meet ψ . Predicate $\psi_1 \Rightarrow \psi_2$ is expressed in terms of disjunction and negation: $m \models (\psi_1 \Rightarrow \psi_2)$ iff $m \not\models \psi_1$ or $m \models \psi_2$.

3 Property Diagram. Properties as Goals and Feedback in Simulation

One of the crucial questions in modelling is the question of correctness of a model. The problem is that the description of a system is usually informal and does not suit to check correctness of the model. However, if formulating desired properties a designer is oriented on the internal reachability graph based model, then the properties can become useful for correctness checks. This orientation of designers should be supported by a tool.

Consider again our model of a distribution center (Figure 1). We require that each order (i, c) has to be received by its customer as a finite set of deliveries and the sum of quantity of all deliveries should be equal to the quantity of the order. If a designer will formulate this property looking at the CPN (Figure 1)

he/she will choose only two places *order* and *delivery* to set a relation between each token in place *order* and a set of tokens in place *delivery*:

Property of an accepted order delivery:

$$\forall i [i : STRING], \forall c [c : INT] \\ (AG(order.(i, c) \Rightarrow \\ AF(delivery.(i_1, c_1), \dots, (i_m, c_m) \\ [i_1 = \dots = i_n = i \wedge c_1 + \dots + c_n = c \wedge n \in \mathcal{N}]))).$$

Sometimes designers are interested in internal properties of a model. For example, in our model the current debt should be delivered as one delivery, the debt has priority with respect to the current orders. Another requirement is that the stock should be replenished. Let us describe all these properties by formulas.

Property of an accepted debt delivery:

$$\forall i [i : STRING], \forall c [c : INT] \\ (AG(debt.(i, c) \Rightarrow AF(delivery.(i, c) \wedge debt.("", 0))))$$

Property of the priority of the current debt with respect to orders :

$$\forall j \forall r \forall i \forall c \\ (AG((debt.(j, r) [j \neq "" \wedge r \neq 0] \wedge order.(i, c)) \Rightarrow \\ AF(delivery.(j, r) \wedge order.(i, c))).$$

Property of the stock replenishment:

$$AG(stock.d [d = 0] \Rightarrow AF stock.d [d > 0]).$$

Equal values of tokens are represented by the matched names, for example the token in place *debt.(j, r)* has the same value as token in place *delivery.(j, r)*.

Analyzing properties we can notice the general pattern of our properties. Each property contains a specification of a start marking, namely, some places and the predicates about tokens in those places. Each property contains also a specification of a goal marking. The relation between the start and the goal markings is represented by a predicate or by matching variables. On the basis of this pattern, the property specification process is realized as a tool wizard, which can be included as an Add-In into any existing CPN-tool. In a separate window the tool shows to designers the specification steps and builds a property diagram corresponding to a property specified by the designers.

Property Diagram. We define a property diagram for the CPN as a UML class diagram [7] with two sets of classes and two types of associations

$$PD = (M, T, MT, MM),$$

- M is a set of classes of stereotype $\ll Marking \gg$. For example, classes *Start* and *Goal* in Figure 4.
- T is a set of classes of stereotype $\ll Token \gg$. The name of a token is the name of one of places from the corresponding CP-net. Each token possesses a set of attributes.

For instance, token *order* in Figure 4 has an identifier i and a quantity c as its attributes.

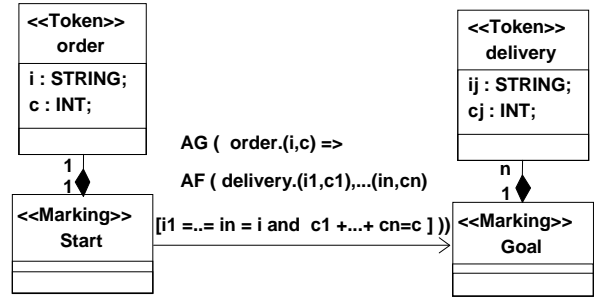


Figure 4: Property diagram

- $MT = \{(marking, token) | marking \in M, token \in T\}$ is a composition relation. Each element of this relation $(marking, token)$ is shown by an association arrow with the filled diamond end. The diamond is filled because we do not consider empty markings for property specification. Several tokens of one color are presented by multiplicity of the composition.

For example, marking *Goal* is composed from n tokens *delivery*.

- $MM = \{(marking_1, marking_2) | marking_1, marking_2 \in M\}$ is a relation on markings. It is represented with an association arrow, labelled by a property formula (Figure 4).

Let a CP-net be given. The functions of the tool wizard:

- **Composing the start marking**
 1. The tool suggests a designer to point out a place for the start marking at the CP-net.
 2. The designer chooses the place.
 3. The tool draws a box representing the token of the chosen place and shows the list of variables used in the CP-net to represent the token.
 4. The designer chooses variables for the property specification. For example, in Figure 1, a token in place *delivery* is represented by pairs (i, c) , (i, r) or $(i, d - r)$. We have chosen the pair (i, c) for specification of the property represented by Figure 4.
 5. Steps 1,2,3 are repeated if the start marking contains more than one place.
 6. The tool draws a box of stereotype $\ll Marking \gg$ and the composition relation with the chosen tokens.
 7. The tool suggests to specify the multiplicity of the composition relation with each of chosen tokens.
 8. The designer specifies the multiplicity.
- **Composing the goal marking**

The same steps 1 to 8 are executed to compose the goal marking.

- **Specification of a property as an association between the start and the goal markings**

1. The tool draws the association arrow between the start and the goal markings at the property diagram and recommends to specify the association label using lists of property patterns, allowed variables defined at the diagram and some logical symbols.
2. The designer specifies the association label.

Other property pattern uses temporal quantifier EF to specify the existence of a path of the reachability graph with specific properties.

The property specification can be used to increase the confidence about the correctness of a CP-net. For models with finite reachability graphs we can prove correctness with respect to a property. Models with infinite reachability graphs can be simulated with property checks. Simulation does not allow verifying the model completely, but each random simulation run can indicate possible mistakes.

A simulation sequence in the property oriented simulation is a reachability graph path, for which we check a property. A simulation sequence $a, deliver, b, partly, c, supply, d, rest, f$ is highlighted in Figure 2. A property will be checked as follows:

1. For each node of a simulation sequence the start predicate of the property is checked.

For example, the start predicate of the *Property of an accepted order deliveries* is specialized into two predicates: $order.(^{\prime}A^{\prime}, 40)$ and $order.(^{\prime}B^{\prime}, 300)$ and both predicates hold in node a (Figure 2).

2. If the node, where the start predicate holds, has not been found then the model is incorrect with respect to the specified property.

3. If the node, where the start predicate holds, has been found, then for the next nodes of the simulation sequence the goal predicate of the property is checked.

The goal predicate for our example is transformed into two predicates $delivery.(^{\prime}A^{\prime}, c_1)...(^{\prime}A^{\prime}, c_n) [c_1 + \dots + c_n = 40]$ and $delivery.(^{\prime}B^{\prime}, c_1)...(^{\prime}B^{\prime}, c_n) [c_1 + \dots + c_n = 300]$.

4. If the node, where the goal predicate holds, has been found, then the complete property holds and the simulation sequence is stopped and indicated as a sequence for which the property holds.

For example, the first of the goal predicates holds in node b , where $delivery.(^{\prime}A^{\prime}, 40) [40 = 40]$, and the second one - in node f , where $delivery.(^{\prime}B^{\prime}, 160), (^{\prime}B^{\prime}, 140) [160 + 140 = 300]$.

5. If the node, where the goal predicate holds, has not been found, then this indicates one of two possible situations: either the property does not hold or the simulation has been stopped too early. To find out the answer an analyst can extend the simulation sequence and scan it with a start conditions corresponding to the goal predicate the correctness of which has not been proved.

For example, if the simulation sequence will be stopped in node c , predicate $delivery.(^{\prime}B^{\prime}, c_1)...(^{\prime}B^{\prime}, c_n) [c_1 + \dots + c_n = 300]$ will not hold and the analyst will extend the simulation sequence and find the node f where the property holds.

The logic for CPN property specification allows classifying useful property patterns and restricts the variety of algorithms for property oriented simulation.

4 Conclusions

In this paper we have proposed a logic for property specification for the Coloured Petri Nets, and found some patterns to represent the properties. We have developed a suitable form to specify properties by property diagrams. The logic which we have proposed classifies formulas and makes it possible to construct specific tool-forms to specify properties. The elements of the properties should be chosen by a designer from a CP-net.

The specified properties can be checked during the first round of simulation aimed at correctness checks of the model. If the properties hold for all specified tokens and paths, then the designers can confidently measure other parameters of the model during the next round of simulation.

In future, we are going to investigate formulating and visualizing properties for the CPN with time stamps and for the hierarchical CPN. Another promising subject is relating some classes of properties to correct abstractions from colors.

References

- [1] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, Volume 1*, Springer-Verlag, 1997.
- [2] J.L.Peterson, *Petri Net theory and the modeling of systems*, Englewood Cliffs : Prentice-Hall,1981.
- [3] A. Kindler, T.Vesper, *ESTL:A Temporal Logic for Events and States*, In: Desel, J.; Silva LNCS 1420, ICATPN'98, Springer-Verlag, 1998, pp. 365-384.
- [4] E.A.Silver, D.F.Pyke, R.Peterson, *Inventory Management and Production Planning and Scheduling*, John Willey and Sons, 1998.
- [5] R. Alur and C. Courcoubetis and D.L. Dill, *Model-Checking in Dense Real-Time, Information and Computation*, Vol. 104(1), pp. 2-34,1993.
- [6] Z.Manna, A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems. V.1. Specification*, Springer-Verlag, 1992.
- [7] OMG, UML2. <http://www.omg.org/uml>, 2003.