DECENTRALISED ACTIVE CONTROLLER

Chiheb Ameur Abid and Belhassen Zouari
LIP2 Laboratory, University of Tunis, El Manar 2, Tunis, Tunisia

Keywords: Supervisory control theory, Coloured Petri nets, Discrete event systems, Distributed systems.

Abstract: This paper deals with the synthesis problem of controllers based on Coloured Petri nets in a decentralized setting. An approach is proposed to derive a decentralized controller consisting of a set of local controllers. Each local controller, modeled by a CP-net, observes and controls a portion of the plant model. Communication between local controllers is performed only when it is necessary allowing the appropriate local controller to restrict the behavior of its observed portion in order to meet the control specification.

1 INTRODUCTION

The supervisory control theory for discrete-event systems (DES) aims to control a plant model by the automatic synthesis of a controller (Ramadge and Wonham, 1989). Due to their easily understood graphical representation and well-formed mathematical formalism, Petri nets have been widely used to investigate this theory (Giua and DiCesare, 1994; Holloway et al., 1997; Ghaffari et al., 2003).

In practice, implementing a centralised controller is not always feasible since a plant model can be a collection of several semi-autonomous components geographically dispersed. As a solution, the decentralised control (Rudie and Wonham, 1992) allows to looks for a fixed number of controllers such that each controller observes and controls a portion of the plant. The decentralised control can be performed with communication (Guan and Holloway, 1995), without communication (Basile et al., 2007), or by introducing a central coordinator (Chen and Baosheng, 1991).

Although, they seem to be powerful enough to describe complex systems in a manageable way, and their expressiveness capabilities, few works have used Coloured Petri nets (CP-nets) (Jensen et al., 2007) to investigate the control problem. In (Makungu et al., 1999), the problem of forbidden states is tackled for controlled CP-nets. In our previous works (Abid and Zouari, 2008; Abid et al., 2010), we have proposed a first approach combining both theory of regions and CP-nets in which the obtained controller is reduced to one place. Also, we have optimised the same approach for symmetric systems to tackle the state explosion problem. In a second approach, we generate a CP-net controller as an active process without using the theory of regions.

The work presented in this paper addresses the supervisory control problem in a decentralised setting. The considered plant model consists of several modules, modelled by CP-nets, which communicate through the synchronisation of common actions. Based on global control specifications expressed in terms of forbidden states, we propose to build a *decentralised active controller*. Such a controller consists of a set of communicating *local controllers* associated with modules. The role of a each local controller is to observe and to control its associated module.

The remainder of the paper is organised as follows. Section 2 introduces the CP-net models. Section 3 provides the computation of necessary information related to the building of a decentralised active controller. Section 4 deals with the building of a CP-net model representing a local controller. Finally, section 5 summarizes the main conclusions and perspectives of this work.

2 PRELIMINARIES

In this paper, the plant model to be considered is assumed to be made up of several communicating modules. The communication is ensured via the execution of common actions. Further, we assume that each module corresponds to a *Resource Allocation System* (RAS). A RAS, as defined in (Reveliotis et al., 1997), corresponds to a set of resources, that can be of many

Ameur Abid C. and Zouari B. (2010).
DECENTRALISED ACTIVE CONTROLLER.
In *Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics*, pages 252-259
DOI: 10.5220/0002946102520259

types and available in any amount of copies, and a set of generic processes, that can be instantiated in a collection of processes.

Formally, a RAS is a tuple $(R, GP, GetR, PutR)$ where:

- $R = (RT, Rnb)$ is the resource specification, s.t.RT is the finite set of resource types, and $Rnb \in Bag(RT)$[1] is a multiset on RT representing the amount of available copies of each resource type.
- GP is the finite set of generic processes.
  A generic process $Gi =< P_{si}, F_i, C_i >$ is defined by a finite state machine (FSM) as follows:
  - $P_{si} = \{p_{0i}, p_{i1}, p_{i2}, ..., p_{il(s)}\}$ is the finite set of states, where p_{0i} represents the 'idle' place of the generic process. The 'idle' place is the initial place of whole process instance.
  - $F_i : P_{si} \rightarrow P_{si}$ is the flow relation representing the state transitions.
  - C_i is the finite set of identities of process instances.
- $GetR : F_i \rightarrow Bag(R)$ is the resource allocation function
- $PutR : F_i \rightarrow Bag(R)$ is the resource restitution function

A RAS can be modelled by a CP-net. Such a CP-net is built from a collection of FSMs (representing the generic processes), a resources place, and connection arcs related to the resource management.

A marked CP-net N representing a RAS, where $N =< P, T, C, W^-.W^+, \Phi, M_0 >$, is defined as following:

- $P = P_S \cup \{p_R\}$ is a finite set of places where $P_S = \bigcup_{i=1}^{|GP|} P_{si}$, and p_R represents the resources place such that $P_{si} \cap P_{sj} = \emptyset$ and $P_S \cap \{p_R\} = \emptyset$.
- T is a set of transitions recursively built as follows:
  Initially $T = \emptyset$, then $T = T \cup \{f_{ij}\}; i = 1, .., |GP|; j = 1, .., |F_i|$.
  The elements of T may be lexicographically renamed so as we can use the notation $T = \{t_i\}; i = 1, .., |T|$.
- $C = \bigcup_{i=1}^{|GP|} C_i \bigcup_{i=1}^{|GP|} (C_i \times C_r) \cup C_r$ where C_r is a colour domain of $p_R, C_r = RT$.
- $W^- = W_S^- \cup W_R^-$, where $W_S^- : (P_s \times T) \rightarrow \{0, \bigcup_{i=1}^{|GP|} X_i\}$ such that X_i is a variable defined on C_i, and $W_R^- : (\{p_R\} \times T) \rightarrow Bag(C_r)$ such that $\forall t \in T, W_R^-(p_R, t) = GetR(f_{ij})$.
- $W^+ = W_S^+ \cup W_R^+$, where $W_S^+ : (P_s \times T) \rightarrow \{0, \bigcup_{i=1}^{|GP|} X_i\}$ such that X_i is a variable defined on C_i, and $W_R^+ : (\{p_R\} \times T) \rightarrow Bag(C_r)$ such that $\forall t \in T, W_R^+(p_R, t) = PutR(f_{ij})$.
- M_0 the initial marking is a function defined on P, such that $M_0(p) = Bag(C(p))$, for all $p \in P$. M_0 is defined as follows:
  $\forall p \in P_S \setminus (\bigcup_{i=1}^{|GP|} \{p_{0i}\}); M_0(p) = 0; M_0(p_R) = Rnb$ and $M_0(p_{0i}) \geq 1$.
- Φ is a function which associated a guard with any transition. By default $\Phi(t)$ is *true* for any transition t.

We consider a decentralised system consisting of several modules that communicate. We assume that each module corresponds to a RAS modelled by a CP-net. The communication between CP-nets can be performed through two possible forms. The asynchronous form by means of shared places, or in synchronous form by means of shared transitions. Since communication via shared places can be transformed into synchronisation via shared transitions, it suffices to support the latter (Christensen and Petrucci, 2000). Modular CP-nets (Christensen and Petrucci, 1995) allow to model efficiently such systems. The latter models introduce structure to CP-nets by letting modules specified separately. For further details about modular CP-nets, one may refer to original papers (Christensen and Petrucci, 1995).

Let be $K = \{1, ..., m\}$. A modular CP-net is a pair $MCPN = (\{N_k | k \in K\}, TF)$ satisfying:

- $\{N_k | k \in K\}$ is a set of modules such that $\forall k \in K, N_k =< P_k, T_k, C_k, W_k^-.W_k^+, \Phi_k >$ is a CP-net,
- $TF \subseteq 2^T$ is a finite set of transition fusion sets where $T = \bigcup_{k \in K} T_k$.

In a modular CP-net, all transitions belonging to the same transition fusion set are fired as one indivisible action sharing the values assigned by a common *binding*. The binding is a function used to assign only one value for a variable of a transition fusion set, i.e. a variable name refers to the same value for all transitions in a transition fusion set.

Example 1. *Throughout this paper, we consider the producer-consumer problem. In this problem, there are two kinds of processes, namely producers and consumers, sharing a stock (a resource). Figure 1(a) gives a CP-net modelling this problem.* $C_1 = \{pr\}$, $C_r = \{f, o\}$ *and* $C_2 = \{co\}$ *are the colour classes of this net. A token pr is used to model the behaviour of a producer, while a token co allows to model the behaviour of a consumer. The token f indicates the number of free places in the stock, while the token o indicates the produced objects. We assume that the stock capacity is two. The colour domains of places are:* $C(a1) = C(a2) = C_1$, $C(r) = C_r$ *and*

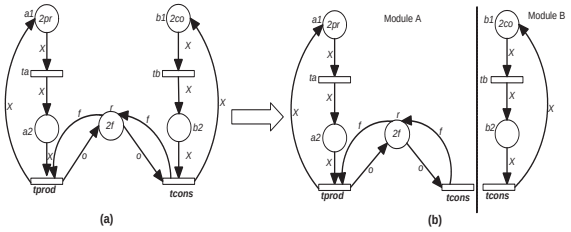

Figure 1: Example of CP-nets.

$C(b1) = c(b2) = C_2$.
As it is illustrated in Figure 1, the latter CP-net can be decomposed into two modules A and B. These modules are both synchronised on transition tcons. The two shared transitions tcons of module A and B constitute a transition fusion set.

3 DECENTRALISED ACTIVE CONTROLLER MODEL

In this section, we present the synthesis approach of a decentralised active controller. As a plant model, we consider a decentralised DES consisting of several modules modelled by a modular CP-net. We assume that every module is structured on a set of generic processes sharing a set of resources. As our aim is to resolve the state forbidden problem, the control specification is then expressed in terms of *forbidden markings*. In a first step, we use the reachability graph to determine the *dangerous markings* as well as transitions to be disabled from firing by the local controllers, called the *forbidden transitions*. The dangerous markings are markings from which the control actions must be applied by disabling some forbidden transitions.

3.1 Architecture of a Decentralised Active Controller

A decentralised controller consists of a set of local controllers where each one is associated with one module of the plant model. The preventing of a transition from firing in a dangerous marking is ensured by the local controller associated with the module to which the transition belongs. Each local controller has to observe permanently the evolution of its associated module by tracking its current local marking. However, since a control action depends mainly on the reached global state of the plant model (the entire system), then a local controller requires to observe some evolutions of the plant model. Such a global observation is performed, when it is necessary, by communicating with the other local controllers. In summary, each derived local controller associated with a module has three main features:

- Observing the current local state of its module,
- Communicating with the other local controllers to determine the global state of the plant model when it is necessary,
- Performing control actions by preventing the firing of some transitions of its associated module.

For sake of simplicity, we propose to decompose a local controller into two submodules. The first one, called *communication submodule*, observes and diffuses the current local marking of its associated module, while the second submodule, called *executor submodule*, performs the control actions by preventing some transitions of its associated module from firing. For this purpose, the executor submodule must be able to identify when the overall plant model reaches a dangerous marking. This ability is ensured by communicating with communication submodules. It is worth noting that an executor submodule is associated with a module only when there exists at least one transition of the module to be disabled by the controller.

3.2 Determining the Admissible Behaviour

The supervisory control theory classifies the transitions into two categories. First category consists of the *controllable transitions* which may be disabled when it is necessary. In contrast, the transitions belonging to the second category, called *uncontrollable transitions*, are beyond any control procedure. Hence, we assume that the transition set of the plant model is partitioned into two disjoint subsets: the set of controllable transitions and the set of uncontrollable ones.

The disabling of a transition is performed in a *dangerous marking* from which the firing of the transition leads to a forbidden marking. So that, we have to identify the set Ω of *state-transitions* to be disabled. Every element of Ω is a couple $(M,(t,c))$ where M is a dangerous marking, and (t,c) is a controllable coloured transition, called *forbidden transition*, such that the firing of t with colour c from M yields to a forbidden marking.

The set Ω of state-transitions is computed from the reachability graph of the plant model. The basic idea is to remove forbidden nodes. Nodes becoming unreachable from the initial marking and the non coreachable markings must be removed from the admissibility graph. The identification of dangerous and

forbidden nodes is performed according to the following rules:

- a marking is qualified as dangerous if it has at least one output arc where its destination is a forbidden marking,
- a marking is qualified as forbidden when it has no output arcs and it is not a final marking, or it is a dangerous marking and it has at least one output arc labelled by an uncontrollable transition,
- every forbidden marking must be removed with its input and output arcs.

In (Abid et al., 2010), an algorithm is proposed for the computation of the admissibility graph and the set Ω of state-transitions.

Example 2. *Consider the example of producer-consumer. Its reachability graph is illustrated by Fig. 2. Assume that we want to restrict the stock capacity to hold only one object. Therefore, the initially specified set of forbidden markings is*

$$FM = \{M10, M13, M14, M15, M16, M17\}.$$

By determining the admissible behaviour, we obtain the admissibility graph of Fig. 3 and the set of state-transitions $\Omega = \{(M6, (tprod, < pr, o >)), (M11, (tprod, < pr, o >)), (M12, (tprod, < pr, o >))\}$, *where* $M6 =< 1,1,1,1,1,0 >$, $M11 =< 1,1,1,1,0,1 >$ *and* $M12 =< 0,2,1,1,0,1 >$. *Thus, the coloured transition* $(tprod, < pr, o >)$ *has to be disabled in markings* $M6$, $M11$ *or* $M12$.

Let d be a dangerous state. We use $FT(d)$ to denote the set of coloured forbidden transitions related to d. Each element of $FT(d)$ is a coloured transition (t,c) such that the firing of (t,c) in d leads to a forbidden or inadmissible state. Let k be a module, we use FT_k to to determine the set of forbidden transitions of module k. Let DS be the set of dangerous states. FT may be viewed as an application defined as follows:

$$\begin{aligned} FT: \quad & DS \rightarrow 2^T \\ & d \mapsto \{(t,c) | (d,(t,c)) \in \Omega\} \end{aligned}$$

Basically, a local controller can only observe the behaviour of its associated module. Since each local controller has to communicate with the other local controllers in order to build and identify dangerous markings, we introduce the notion of *dangerous local marking*. A dangerous local marking of a module is a local marking from which we can build a dangerous marking.

Let M_k be a local marking of module k.
M_k is a local dangerous marking iff $\exists M' \in [M_0\rangle$: $M'_k = M_k \wedge M'$ is a dangerous marking.

For every local marking associated with a module k, we make available the following data:

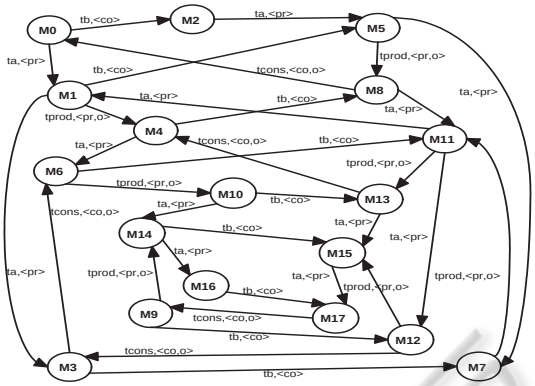

Figure 2: Example of a reachability graph.

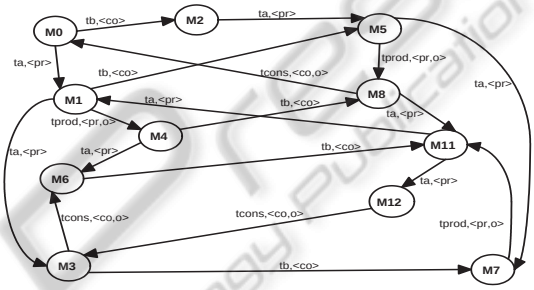

Figure 3: Example of admissibility graph.

- the set of dangerous markings,
- the set of dangerous local markings of module k,
- the set FT_k of forbidden transitions of module k and their correspondent dangerous marking.

Example 3. *From the set* Ω *computed in previous example, we determine the following sets:*
The set of dangerous local markings of module A is $SDLM_A = \{< 1,1,1,1 >, < 0,2,1,1 >\}$.
In module A, there exists only one forbidden transition:
$FT_A(M6) = FT_A(M11) = FT_A(M12) = \{(tprod, < pr, o >)\}$.
The set of dangerous local markings of module B is $SDLM_B = \{< 1,0 >, < 0,1 >\}$.
In module B, there is no forbidden transitions, then: $FT_B(M6) = FT_B(M11) = FT_B(M12) = \emptyset$.

4 A LOCAL CONTROLLER MODEL

Now, we describe the CP-net model of a local controller. The key idea of the decentralised active controller approach is that a local controller must be able to detect the reaching of a dangerous state. Once a

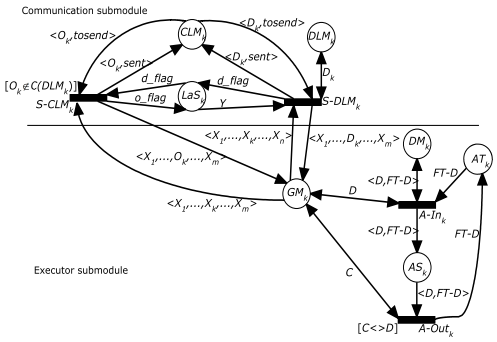

Figure 4: CP-net modelling a local controller.

dangerous marking is detected, the local controller removes appropriate authorisations in order to disable the firing of some forbidden transitions of its associated module. Basically, a local controller has to handle, permanently, information about the current local marking of its module. In contrast, a global marking is determined for a local controller through communication. Intuitively, a local controller diffuses its observed local state to the other local controllers when it detects that the observed local state may be a part of a dangerous global state.

As precited, a local controller can be seen as the composition of two submodules, namely the communication and the executor submodules. The main purpose of the first submodule is to determine and diffuse the current local marking of its associated module to other local controllers, while the executor submodule performs the control actions. More precisely, the communication submodules send the determined local markings to every executor submodule. Then, each executor module can determine whether the plant model reaches a dangerous global marking from which some transitions have to be prevented from firing.

The communication submodule diffuses the current local state of its associated module according to the following rules:

- Its associated module reaches a dangerous local state from another local state (dangerous or not),
- Its associated module reaches a non dangerous local state from a dangerous local state.

Figure 4 describes the CP-net modelling a local controller. Let us consider a DES, made up of m modules, modelled by a modular CP-net $MCPN = (\{N_k | k \in K = \{1,...,m\}\}, TF)$, where $\forall k \in K, N_k =< P_k, T_k, C_k, W_k^-, W_k^+, M_{0,k}, \Phi_k >$. In a first step, we detail the formal definition of the communication submodule. Secondly, we define the executor CP-net submodule.

Step 1: the communication submodule. Let us consider the CP-net N_k modelling module $k \in K$. The model representing module k connected to its associated communication submodule is a CP-net $N1_k$ obtained from N_k, so that $N1_k =< P1_k, T1_k, C1_k, W1_k^-, W1_k^+, M1_{0,k}, \Phi 1_k >$.
$P1_k = P_k \cup \{CLM_k, DLM_k, LaS_k\}$, where:

- CLM_k represents the current local marking of module k,
- DLM_k contains the set of of dangerous local markings associated with module k,
- LaS_k indicates whether the last communicated current local marking of module k is dangerous, or not.

$T1_k = T_k \cup \{S-DLM_k, S-CLM_k\}$, where:

- $S-DLM_k$ is a synchronised transition allowing to send a dangerous local marking,
- $S-CLM_k$ is a synchronised transition. It allows to send a non dangerous current local marking when it is reached from a dangerous global marking.

Now, we introduce the following additionnal colour classes:
$C_{num} = \{1,2,\ldots,MaxInt\}$ is a colour class representing a set of finite positive integers. The elements of C_{num} are used to model the occurrences of some given tokens. We assume that $MaxInt$ is large enough to be greater than the bound of maximum occurrences of any token in a reachable marking,
$C_{status} = \{tosend, sent\}$ is a colour class where the object *sent* (resp. *tosend*) is used to identify whether a current local marking is already sent to all executor submodules (resp. not yet sent),
$C_{flag} = \{d\_flag, o\_flag\}$ where the object d_flag (resp. o_flag) is used to indicate whether the last diffused local marking is dangerous (resp. not dangerous).
Every added place is characterised by a colour domain and an initial marking defined as follows:

- Place LaS_k allows to hold information indicating whether the last communicated current local marking is dangerous or not. For this purpose,we use the colours of C_{flag}. Thus, $C(LaS_k) = C_{flag}$. Initially, place LaS_k is empty.
- Place CLM_k holds the current local marking of module k, and information whether this local marking is communicated. A local marking is represented by a tuple made up of counters. Each counter holds the information about the occurrence of tokens in a given place (according to the lexical order) among process places and the resource place. Such a representation is obtained

through a Cartesian product performed on the basis of the number l of process classes in module k, the number of places x_i per a process i and the resource class $C_{r,k}$. More precisely, a local marking is an element of the following colour class:

$$C_{LM,k} = \bigotimes_{i=1}^{l} \bigotimes_{j=1}^{x_i} C_{num} \bigotimes_{u=1}^{|C_{r,k}|} C_{num}.$$

In addition to the current local marking, place CLM_k holds information indicating whether the current local marking is already communicated or not. It is performed by using elements of the colour class C_{status}. Thus, the domain class of place CLM_k is $C(CLM_k) = C_{LM,k} \times C_{status}$.
Place CLM_k is always mono-marked and its initial marking $M_{0,k}(CLM_k)$ is determined from the initial marking of module k.

- Place DLM_k holds the set of dangerous local markings of module k. Each marking is represented in identical way as for place CLM_k. So that $C(CLM_k) = C_{LM,k}$. Initially, place DLM_k holds the set $SetDLM_k$ which represents the dangerous local markings of module k, i.e. $M_{0,k}(DLM_k) = SetDLM_k$. This place is only read accessed.

Now, we give the colour functions associated with the arcs connecting places to transitions.

- As place CLM_k holds the current local marking of module k modelled by CP-net N_k, then it has to be connected to every transition of T_k in order to keep its information up to date. This is because the firing of every transition of T_k may update the current local marking of module k. Place CLM_k is connected to every transition of T_k with an input arc (reading marking) and output arc (updating marking) as follows:
$\forall t \in T_k$,
$W1_k^-(CLM_k,t) =< X_{1,1},\ldots,X_{i,j},Y_1,\ldots,Y_u >= < X >$, where $X_{i,j}$ is a variable defined on C_{num} computing the number of tokens in place i of the process type j ($p_{ij} \in P_k$), and Y_u is a variable defined on C_{num} reading the occurrence of colour u in resource places.
$\forall t \in T_k$,
$W1_k^+(CLM_k,t) =< X'_{1,1},\ldots,X'_{i,j},Y'_1,\ldots,Y'_u >= < X >$, where $X'_{i,j}$ and Y'_u are variables defined on C_{num} and determined as follows:
$X'_{i,j} = X_{i,j} - \chi$, where $\chi = W^+(p_{ij},t) - W^-(p_{ij},t)$,
$Y'_u = Y'_u - \xi$, where ξ is computed as follows:

$$\begin{cases} W_k^-(r,t) = \sum_i \alpha_i.r_i, \\ W_k^+(r,t) = \sum_i \alpha'_i.r_i \end{cases} \Rightarrow \xi = \alpha'_u - \alpha_i.$$

- The colour functions related to $S-DLM_k$:
$W1_k^+(DLM_k, S-DLM_k) = W1_k^-(DLM_k, S-DLM_k) =< D_k >$;
$W1_k^-(CLM_k, S-DLM_k) =< D_k, tosend >$;
$W1_k^+(CLM_k, S-DLM_k) =< D_k, sent >$;
$W1_k^-(LaS_k, S-DLM_k) =< Y >$;
$W1_k^+(LaS_k, S-DLM_k) =< d\_flag >$;
where $D_k \in C_{LM,k}$ and $Y \in C_{flag}$.
The colour functions related to $S-CLM_k$:
$W1_k^-(CLM_k,, S-CLM_k) =< O_k, tosend >$;
$W1_k^+(CLM_k,, S-CLM_k) =< O_k, sent >$;
$W1_k^-(LaS_k, S-CLM_k) =< d\_flag >$;
$W1_k^+(LaS_k, S-CLM_k) =< o\_flag >$;
where $[O_k \in C_{LM,k}]$.
Transition $S-CLM_k$ is associated with the guard $[O_k \notin SetDLM_k]$.

Step 2: the executor submodule. Now, we present the features of the executor submodule. It is obtained from the CP-net $N1_k =< P1_k, T1_k, C1_k, W1_k^-, W1_k^+, M1_{0,k}, \Phi 1_k >$, modelling module k with its associated communication submodule.
The model representing a module k and its local controller is a CP-net N_k^* obtained from $N1_k$, so that $N_k^* =< P_k^*, T_k^*, C_k^*, W_k^{*-}, W_k^{*+}, M_{0,k}^*, \Phi_k^* >$, where:
$P_k^* = P1_k \cup \{GM_k, DM_k, AS_k, AT_k\}$, where:

- GM_k represents the global marking of the overall plant model,
- DM_k holds the set of dangerous global markings,
- AS_k represents the alert state of the local controller associated with module k,
- AT_k contains the set of authorisations for forbidden transitions of module k.

$T_k^* = T1_k \cup \{A-IN_k, A-OUT_k\} \bigcup_{i \in K \setminus \{k\}} \{S-DLM_{k,i}\} \bigcup_{i \in K \setminus \{k\}} \{S-CLM_{k,i}\}$, where:

- $A-IN_k$ allows to enter the alert state,
- $A-OUT_k$ allows to quit the alert state,
- $\bigcup_{i \in K \setminus \{k\}} \{S-DLM_{k,i}\}$ is a set of shared transitions. The firing of a transition $S-DLM_{k,i}$ allows the executor submodule k to receive a diffused dangerous local marking from the communication submodule i.
- $\bigcup_{i \in K \setminus \{k\}} \{S-CLM_{k,i}\}$ is a set of shared transitions. The firing of a transition $S-CLM_{k,i}$ allows module k to receive a non dangerous local marking from the communication submodule i.

The colour domain and the initial marking associated with every added place are as follows:

- Place GM_k holds the marking of the overall plant model. Its colour domain is a Cartesian product of colour classes representing local markings of individual modules. Thus, the colour domain of GM_k is defined as follows:

$$C^*(GM_k) = \prod_{i \in K} C_{LM,i}$$

  Initially, the marking of place GM_k represents the initial marking of the overall plant model.

- Place DM_k maintains the set of state-transitions of module k. Its colour domain is

$$C_k^*(DM_k) = C^*(GM_k) \times C_{FT,k}$$

  , where $C_{FT,k}$ denotes the colour class representing forbidden transitions of module k. This place is only read accessed.

- Place AS_k indicates that the controlled module k is in alert state. Its marking indicates the global marking of the controlled system and the transition to be prevented from firing.

$$C_k^*(AS_k) = C^*(GM_k)$$

  Initially, place AS_k is empty as all forbidden transitions are authorised.

- Place AT_k is connected to every forbidden transition by one input/ output arc in order to check the presence of the associated firing authorisation: $\forall t \in FT_k, W_k^{*+}(AT_k,t) = W_k^{*-}(AT_k,t) =< Xt >$, where Xt is defined on $C_{FT,k}$ and represents the identity of the coloured forbidden transition. Further, we associate with every forbidden transition the following guard expression:

$$[\bigvee_{c \in C(t)} [\bigwedge_{X \in Var(t) \setminus \{Xt\}} X = X(c) \wedge Xt = id\_ft(t,c)]]$$

  where id_ft is a function which maps a forbidden coloured transition (t,c) to an element of $C_{FT,k}$ representing the identity of (t,c). This associated guard expression allows to associate the appropriate authorisation for a coloured transition.

The colour functions of the arcs connecting the controller places to a subset of T_k and to the transitions $A-In_k$ and $A-Out_k$ are defined as follows:
$W_k^{-*}(DM_k, A-In_k) = W_k^{*+}(DM_k, A-In_k) =< D, FT-D >$
$W_k^{-*}(GM_k, A-In_k) = W_k^{*+}(GM_k, A-In_k) =< D >$
$W_k^{-*}(AT_k, A-In_k) =< FT-D >$
$W_k^{+*}(AS_k, A-In_k) =< D, FT-D >$
$W_k^{+*}(AS_k, A-Out_k) =< D, FT-D >$
$W_k^{-*}(GM_k, A-Out_k) =< C >$
$W_k^{+*}(AT_k, A-Out_k) =< FT-D >$.

D and $C \in C_{GM}$, $FT-D \in C_{FT,k}$.
Transition $A-Out_k$ is associated with the predicate $[C \neq D]$. For every $i \in K$, place GM_k is connected to the $S-DLM_{k,i}$ and $S-CLM_{k,i}$ allowing to update the local marking of module i in place GM_k. The associated colour functions are defined as follows:
$W_k^{-*}(GM_k, S-DLM_{k,i}) =< X_1, ..., X_i, ..., X_m >$;
$W_k^{+*}(GM_k, S-DLM_{k,i}) =< X_1, ..., X_{i-1}, D_i, ..., X_m >$;
$W_k^{-*}(GsM_k, S-CLM_{k,i}) =< X_1, ..., X_i, ..., X_m >$;
$W_k^{+*}(GM_k, S-CLM_{k,i}) =< X_1, ..., X_{i-1}, D_i, ..., X_m >$.
In summary, the controlled model, obtained from *MCPN*, is modelled by the modular CP-net $MCPN^* = (\{N_k^* | k \in K\}, TF^*)$, where $TF^* = TF \cup \{FS-CLM_k | i \in K\} \cup \{FS-DLM_k | k \in K\}$, where:

- $\forall k \in K$, $FS-CLM_k$ is a transition fusion set such that $FS-CLM_k = \{S-CLM_{k,i} | i \in K\}$;
- $\forall k \in K$, $FS-DLM_k$ is a transition fusion set such that $FS-DLM_k = \{S-DLM_{k,i} | i \in K\}$.

Remark 1. *The transitions $S-CLM_k$, $S-DLM_k$, $A-In_k$ and $A-Out_k$ must have special high priority over all the transitions of the plant model and are fired immediately when enabled.*

Example 4. *In our considered example, only tprod which belongs to model A is a forbidden transition. Thus, the local controller associated with module A consists of the communication and the executor submodules, while the controller associated with module B consists only of a communication submodule. The different features of the local controller associated with module A are:*
$C_A^*(CLM_A) = C_{num} \times C_{num} \times C_{num} \times C_{num} \times C_{status}$
The first and the second elements respectively define the number of producers in places a1 and a2. The third and the fourth elements respectively handles the occurence number of f and o in place r.
$C_{FT,A} = \{tprodPro\}$ *where tprodPro denotes the authorisation for the firing of transition tprod with colour* $< pr, o >$.
$C_A(GM_A) = C_{num} \times C_{num} \times C_{num} \times C_{num} \times C_{num} \times C_{num}$
where the first four elements allow to handle the local marking of module A, while the fifth and the sixth elements handle the local marking of module B.
The initial markings of places are:
$M_{0,A}^*(CLM_A) =< 2,0,2,0,sent >$
$M_{0,A}^*(DLM_A) =< 1,1,1,1 > + < 0,2,1,1 >$
$M_{0,A}^*(AT_A) = tprodPro$
$M_{0,A}^*(GM_A) =< 2,0,2,0,2,0 >$
$M_{0,A}^*(DM_A) =< 1,1,1,1,1,0 > + < 1,1,1,1,0,1 > + < 0,2,1,1,0,1 >$
$\forall t \in T_A, W_A^{*-}(CLM_A, t) =< X_{1,1}, X_{1,2}, Y_1, Y_2, Z >$
$W_A^{*+}(CLM_A, ta\{pr\}) =< X_{1,1} - 1, X_{1,2} + 1, Y_1, Y_2, tosend >$

$W_A^{*+}(CLM_A, tprod\{< pr, o >\}) =< X_{1,1}+1, X_{1,2}-1, Y_1-1, Y_2+1, tosend >$
$W_A^{*+}(CLM_A, tconsum\{o\}) =< X_{1,1}, X_{1,2}, Y_1+1, Y_2-1, tosend >$.

5 CONCLUSIONS

In this paper, we have investigated the problem of designing a decentralised controller based on CP-nets for the forbidden states problem. Considering global specifications, the obtained decentralised controller consists of a set of communicating local controllers, where each one is able to observe permanently the current state of its associated module. Communication between local controllers allows, when it is necessary, a local controller to determine the global state of the entire plant model. Each local controller has a fixed structure whatever the system to control. So that, the synthesis approach is reduced to the determination of the parameters related to each local controller. It is performed mainly by determining the admissible behaviour. Further, this makes the proposed approach easier to understand, and to implement in practice.

REFERENCES

Abid, C. A., Zairi, S., and Zouari, B. (2010). Supervisory control and high-level petri nets. In Pawlewski, P., editor, *Petri Nets: Applications*, chapter 14, pages 281–306. INTECH, Vienna, Austria.

Abid, C. A. and Zouari, B. (2008). Synthesis of controllers using symbolic reachability graphs. In *Proceedings of 9th International Workshop of Discrete Event Systems (WODES'08)*, pages 314–321, Göteborg.

Basile, F., Giua, A., and Seatzu, C. (2007). Supervisory control of petri nets with decentralized monitor places. In *26th American Control Conference (ACC07)*, New York, USA.

Chen, H. and Baosheng, H. (1991). Distributed control of discrete event systems described by a class of controlled petri nets. In *Preprints of IFAC International Symposium on Distributed Intelligence Systems*.

Christensen, S. and Petrucci, L. (1995). Modular state space analysis of coloured petri nets. In *Proceedings of the 16th International Conference on Application and Theory of Petri Nets*, pages 201–217, London, UK. Springer-Verlag.

Christensen, S. and Petrucci, L. (2000). Modular analysis of petri nets. *Comput. J.*, 43(3):224–242.

Ghaffari, A., Rezg, N., and Xie, X. (2003). Design of live and maximally permissive petri net controller using the theory of regions. In *Proceedings of IEEE Transactions on Robotics and Automation*, volume 19, pages 137–142, Aarhus.

Giua, A. and DiCesare, F. (1994). Petri net structural analysis for supervisory control. *IEEE Transactions on Robotics and Automation*, 10(2):185–195.

Guan, X. and Holloway, L. E. (1995). Distributed discrete event control structures with controller interactions. In *In proceedings of 1995 American Control Conference*, pages 3151–3156.

Holloway, L. E., Krogh, B. H., and Giua, A. (1997). A survey of petri net methods for controlled discrete eventsystems. *Discrete Event Dynamic Systems*, 7(2):151–190.

Jensen, K., Kristensen, L. M., and Wells, L. (2007). Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *Int. J. Softw. Tools Technol. Transf.*, 9(3):213–254.

Makungu, M., Barbeau, M., and St-Denis, R. (1999). Synthesis of controllers of process modeled as coloured petri nets. *Journal Discrete Event Dynamic Systems Theory Applications Kluwer Academic Publishers*, Vol. 9(No. 2):147–169.

Ramadge, P. and Wonham, W. (1989). The control of discrete event systems. In *Proceedings of IEEE*, pages 81–98. Special Issue on Discrete Event Dynamic Systems.

Reveliotis, S., Lawley, M., and Ferreira, P. (1997). Polynomial-complexity deadlock avoidance policies for sequential resource allocation systems. *IEEE transactions on automatic control*, 42(10):1344–1357.

Rudie, K. and Wonham, W. (1992). Think globally, act locally: Decentralized supervisory control. *IEEE Trans. on Automatic Control*, 37(11):1692–1708.