Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Distributed implementations of Ramadge-Wonham supervisory control with Petri nets

Philippe Darondeau

Abstract— **The purpose of the paper is to assess the benefits of using free labeled and bounded Petri nets as controllers, in the context of Ramadge and Wonham's supervisory control theory. This theory allows, for several types of control problems expressed with regular languages, to decide when they are feasible and to produce finite control automata as solutions. The state graphs of free labeled and bounded Petri nets are a strict subclass of the finite automata. Concentrating on Petri net controllers can only lead to a weaker theory. The compact representation of net controllers is not a definite advantage since modular control synthesis may be used to a comparable effect in the framework of automata and their products. In order to reveal the major benefits of Petri net synthesis for supervisory control, we further impose on nets a structural constraint of distributability. Bounded and distributable nets translate to equivalent systems of finite message passing automata. Distributable net controllers induce therefore distributed control systems for distributed discrete event systems: each DES component receives as its local control component one of the message passing automata, and the locally controlled subsystems interact with one another in fully asynchronous mode. We study in this paper the implementation of Ramadge and Wonham's finite state controllers by distributable net controllers.**

I. INTRODUCTION

Given a finite plant-automaton A with language $\mathcal{L}(A)$ and a regular behavior $B \subseteq \mathcal{L}(A)$, Ramadge and Wonham's theory [1] [2] [3] allows to decide whether $B = \mathcal{L}(A \times C)$ for some finite control automaton C subject to smooth admissibility constraints, where $A \times C$ is the synchronized product of A and C, and it allows to construct such C. Similar procedures apply to the problem $\mathcal{L}(A \times C) \subseteq B$, yielding maximal solutions $\mathcal{L}(A \times C)$ under the assumption that unobservable events are uncontrolable.

Our goal is to search for equivalent Petri net implementations of the finite state controllers produced by Ramadge and Wonham's constructive results. More precisely, we want to decide in which case a finite state controller may be implemented with a bounded Petri net with free labeling (i.e. with injectively labeled transitions). The problem is not trivial: it may occur that $\mathcal{L}(C)$ is not the language of any free labeled and bounded Petri net whenever $B = \mathcal{L}(A \times C)$ for some controller automaton C. The practical interest of free labeled and bounded Petri net controllers may be questioned, precisely since they are weaker than finite automata. A tentative answer is that efficient synthesis procedures exist for free labeled nets, producing compact representations of controllers when they succeed [4]. No comparable procedure is known for the bounded nets with a non injective labeling (therefore equivalent to the finite automata) as the trivial synthesis procedure yields nets with the same size as the corresponding automata. Now the gain in compactness may be deemed an insufficient reward for the lack of generality: compact controllers may be obtained by modular control synthesis [5] without a comparable loss of generality.

In our view, the major benefits of net synthesis appear only after one imposes on nets a structural constraint of *distributability* as follows. A Petri net with a set of transitions $T = T_1 \cup \ldots \cup T_l$, where the T_i's are disjoint, is distributable if its set of places has a similar partition $P = P_1 \cup \ldots \cup P_l$, such that $t \in T_i$ entails $p \in P_i$ whenever transition t consumes tokens from place p. Bounded and distributable nets translate up to divergence-free branching bisimulation to equivalent systems of finite message passing automata $\{C_1, \ldots, C_l\}$, in which no assumption is made on the order nor on the delay of delivery of the messages and the number of messages in transit stays bounded. When the plant automaton A is a system of message passing automata $\{A_1, \ldots, A_l\}$, any distributable net controller may then be turned to a distributed implementation as shown in Fig. 1, where each A_i and C_i interact synchronously while the different *locally* controlled subsystems $A_i \times C_i$ interact asynchronously.

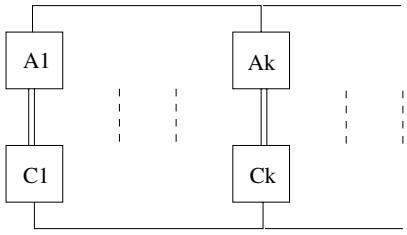

Fig. 1. A distributed controlled system

In the architectures we consider, each control component C_i observes and controls subsets of the corresponding subalphabet T_i. Additionally, different components C_i can send to one another finite sets of messages (not necessarily names of observed transitions). Due to this general communication pattern, the distributed Petri net controllers we propose do not easily compare with the decentralized controllers considered in [6].

Our work has admittedly two main limitations. First, we cannot synthesize *labeled* distributable Petri nets more compact than automata. Second, we cannot synthesize Petri nets from *products* of automata without computing these products, hence we cannot exploit the distributed structure of plant automata: net synthesis occurs downstream from finite

Philippe Darondeau is with INRIA/IRISA, Campus de Beaulieu, 35042 Rennes-Cedex, France (darondeau@irisa.fr)

0-7803-9568-9/05/$20.00 ©2005 IEEE

state controller synthesis, and it takes monolithic controllers as inputs.

The rest of the paper is organized as follows. Section 2 is a brief recall of Ramadge and Wonham's basic results. Section 3 enumerates the problems posed by an adaptation to Petri net controllers. Section 4 presents Petri net synthesis as a tool for controller synthesis. Section 5 brings distributable Petri nets and focusses on their application to distributed control.

II. Ramadge and Wonham's basic theory

The short reminder below is inspired from [1] [2] [3].

A *Discrete Event System* or *DES* is a tuple (Σ, L, L_m) where Σ is a finite alphabet, $L_m \subseteq L \subseteq \Sigma^*$, and L is equal to its prefix closure $\overline{L}$. L_m is the *marked* sublanguage of L. Two independent bi-partitions of the alphabet are assumed, namely $\Sigma = \Sigma_c \cup \Sigma_{uc}$ and $\Sigma = \Sigma_o \cup \Sigma_{uo}$. The event symbols in Σ_c (resp. in Σ_o) are *controlable* (resp. *observable*).

A *controller* is a partial map $f : L \rightharpoonup \mathcal{P}(\Sigma)$ such that $fs \supseteq \Sigma_{uc}$ whenever fs is defined and $fs = fs'$ whenever $\pi_o s = \pi_o s'$, where $\pi_o : \Sigma^* \to \Sigma_o^*$ projects sequences of events to observable subsequences.

The *f-controlled DES* is the tuple (Σ, L^f, L_m^f) where $L_m^f = L^f \cap L_m$ and L^f is the least language containing ε (the empty word) such that $s\sigma \in L^f$ for all $s \in L^f$, $\sigma \in fs$, and $s\sigma \in L$. The controller f is *non-blocking* if $L^f = \overline{L_m^f}$.

A *behavior* $B \subseteq L$ may be *enforced by supervision* ($\vdash B$) if $B = L^f$ for some controller f.

A *marked behavior* $B \subseteq L_m$ may be *enforced by non-blocking supervision* ($\vdash_m B$) if $B = L_m^f$ for some non-blocking controller f.

In order to characterize the behaviors that may be enforced by supervision, Ramadge and Wonham introduce the key concepts of controlability and observability. A sublanguage $S \subseteq L$ is *controlable* if $\overline{S}\Sigma_{uc} \cap L \subseteq \overline{S}$ (the prefix closure of S). It is *observable* if $s\sigma \in L \setminus \overline{S} \Rightarrow s'\sigma \notin \overline{S}$ for all $\sigma \in \Sigma_c$ and for all $s, s' \in \overline{S}$ with equal projections $\pi_o s = \pi_o s'$.

The characterization is as follows.
For $B \subseteq L$, $\vdash B$ *iff* $B = \overline{B}$ and B is controlable and observable.
For $B \subseteq L_m$, $\vdash_m B$ *iff* $B = \overline{B} \cap L_m$ and B is controlable and observable.

Ramadge and Wonham moreover propose sufficient conditions ensuring that whenever some non empty behavior $B' \subseteq B$ can be enforced by supervision, the set of these behaviors has a supremum. This holds in particular if $\Sigma_{uo} \subseteq \Sigma_{uc}$, i.e. when the unobservable events are uncontrolable. In this case the following hold:
if $B = \overline{B} \subseteq L$ then $\vdash \cup\{B' \subseteq B \mid \vdash B'\}$,
if $B = \overline{B} \cap L_m$ then $\vdash_m \cup\{B' \subseteq B \mid \vdash_m B'\}$.

All results above have constructive versions when L and B are languages or marked languages of finite automata. Given $L = \mathcal{L}(A)$ and $B \subseteq L$, one can construct a finite automaton C such that $\mathcal{L}(A \times C) = \cup\{B' \subseteq B \mid \vdash B'\}$ and the induced partial map $f : L \rightharpoonup \mathcal{P}(\Sigma)$ defined with $f(s) = \{\sigma \in \Sigma \mid s\sigma \in \mathcal{L}(C)\}$ for $s \in \mathcal{L}(C)$ is a controller. Similarly, given $L_m = \mathcal{L}_m(A)$ (the language recognized be the final states of A) and $B \subseteq L_m$, one can construct a finite automaton C such that $\mathcal{L}_m(A \times C) = \cup\{B' \subseteq B \mid \vdash_m B'\}$ and f is a non-blocking controller.

It is worth noting that the above control automata C may easily be transformed into equivalent finite automata where all unobservable events induce partial identities on the set of states. It suffices indeed to replace each transition $q \xrightarrow{\sigma} q'$ such that $q \neq q'$ and $\sigma \in \Sigma_{uo}$ with $q \longrightarrow q'$ and $q' \xrightarrow{\sigma} q'$, and to apply determinization using the classical subset construction.

III. Towards Controller Nets

In the sequel, $N = (P, T, F, M_0)$ denotes a Petri net, where $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ defines the weights of the arcs and $M_0 : P \to \mathbb{N}$ defines the initial marking. We assume that the net firing rule is known to the reader. We let $\mathcal{L}(N) \subseteq T^*$ denote the (free) language of N. Since we consider Petri nets with free labeling, we let $T = \Sigma$.

A. Optimal Controller Nets

Let B and L be regular languages, such that $B = \overline{B} \subseteq L = \overline{L}$, and let $S = \cup\{B' \subseteq B \mid \vdash B'\}$. It is assumed throughout the section that either $S = B$ (this holds *iff* B is controlable and observable) or S is regular and $\vdash S$ (this holds whenever $\Sigma_{uo} \subseteq \Sigma_{uc}$). Thus, in all cases under consideration, S (for "supremal") is a regular prefix closed language, and it is controlable and observable. Since $S = L^f$ for some controller f, one may wonder whether $S = L \cap \mathcal{L}(N)$ for some bounded net N "implementing" f. Clearly, the following relations must hold:

$$S \subseteq \mathcal{L}(N) \subseteq S \cup C(L) \tag{1}$$

where C denotes complementation in Σ^*. Now, defining net implementations of controllers is not that obvious.

A first and easy issue is with uncontrolable events. In this respect, N should fulfil the condition stated as:

$$\mathcal{L}(N)\Sigma_{uc} \subseteq \mathcal{L}(N) \cup C(L) \tag{2}$$

Since S is controlable and L is prefix closed, condition 2 follows as a straightforward consequence of condition 1, hence we shall not set it as an explicit requirement on controller nets.

A second issue and problematic is with unobservable events. In order to show the problem, let us consider the automata A and A' shown in figure 2 (the initial state is in black).

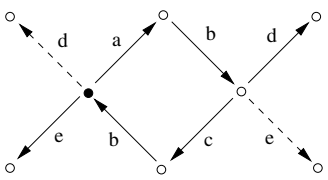

Fig. 2. A (all transitions) and A' (solid transitions only)

Let $L = \overline{L} = \mathcal{L}(A)$, $B = \overline{B} = \mathcal{L}(A')$, and $\Sigma_{uo} = \{a, c\} = \Sigma_{uc}$. The considered behavior B is clearly controlable and observable, hence $S = B$. A Petri net N satisfying condition 1 is shown in figure 3.

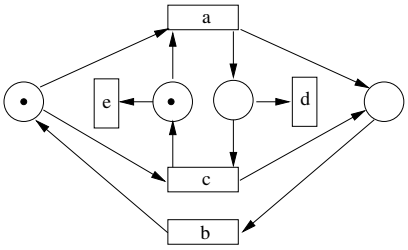

Fig. 3. A tentative net implementation of the controller

It may be verified that $s\sigma \in \mathcal{L}(N) \Leftrightarrow s'\sigma \in \mathcal{L}(N)$ for all $\sigma \in \Sigma_c$ and $s, s' \in \mathcal{L}(N)$ such that $\pi_o s = \pi_o s'$. However, each occurrence of an unobservable event a or c changes the marking of N. It must be the same in every net N satisfying condition 1. Actually, if e and $abcbe$ belong to $\mathcal{L}(N)$ and a and c do not change the markings of N, $M_0[e\rangle$ and $M_2[e\rangle$ at both extremities of some firing sequence $M_0[b\rangle M_1[b\rangle M_2$, and $M_1[e\rangle$ follows by convexity, entailing that $abe \in \mathcal{L}(N)$. We consider that no Petri net implementation of controllers exists in this case. We shall therefore impose on controller nets N the following requirement:

$$(\forall p \in P)\ (\forall \sigma \in \Sigma_{uo})\ F(p,\sigma) = F(\sigma,p) \tag{3}$$

A third and last issue is with the boundedness of controller nets. Imposing on N to be bounded according to the usual definition is overly restrictive, since the firing sequences of N which are not in L cannot be fired anyway in the closed loop system formed of the plant and the controller net. We shall therefore set on controller nets the weaker requirement of boundedness relative to L (L-boundedness) as follows:

$$(\forall p \in P)\,(\exists n)\,(\forall s \in L \cap \mathcal{L}(N))\ M_0[s\rangle M \Rightarrow M(p) \leq n \tag{4}$$

Altogether, a controller net enforcing S should satisfy the conditions 1, 3, and 4.

We look now briefly at marked languages. Let B and L_m be regular languages, with $B = \overline{B} \cap L_m$, and let $S = \cup\{B' \subseteq B \mid \vdash_m B'\}$. Assume that $S = B$ (this holds *iff* B is controlable and observable) or S is regular and $\vdash_m S$ (this holds is $\Sigma_{uo} \subseteq \Sigma_{uc}$). Thus, S is a regular prefix closed language, and it is controlable and observable. Searching for a non-blocking controller net enforcing S amounts to constructing N satisfying conditions 1, 3, and 4 (with $L = \overline{L_m}$).

It will be shown that one can decide whether exist and then construct such controller nets. Note that the parameter S in 1 is a regular language, obtained through the use of Ramadge and Wonham's construction of the "maximal permissive" (finite state) controller. Thus, in the situations considered so far, the synthesis of Petri nets appears as a final phase for implementing finite state controllers produced by independent means.

B. Non Optimal Controller Nets

We consider in this section the cases where the supremal controlable and observable sublanguage S of the "legal" behavior B does not exist or cannot be enforced by controller nets (they are strictly weaker than finite state controllers). Following [1], it is recommended in such situations to specify the "minimal acceptable behavior" B_{min} which is expected.

Given regular languages B and L such that $B = \overline{B} \subseteq L = \overline{L}$ and a regular language $B_{min} \subseteq B$, the question is then to find a controller f such that $B_{min} \subseteq L^f \subseteq B$. An obvious adaptation of this problem is to search for a controller net N satisfying conditions 2, 3, 4 and the relations:

$$\overline{B}_{min} \subseteq \mathcal{L}(N) \subseteq B \cup C(L) \tag{5}$$

Here we cannot dispense with condition 2 because it does not follow from condition 5 (B_{min} is in general not controlable and it may differ from B). Unfortunately, deciding whether there exists some net satisfying the conditions 2, 3, 4, and 5 is an open problem, linked with the synthesis of Petri nets from modal automata [7]. Therefore, in place of 2, we set on controller nets the stronger requirement:

$$(\forall p \in P)\ (\forall \sigma \in \Sigma_{uc})\ F(p,\sigma) = 0 \tag{6}$$

Condition 6 states that no uncontrolable transition consumes tokens from any place, hence it entails $\mathcal{L}(N)\Sigma_{uc}^* \subseteq \mathcal{L}(N)$. If the requirement of boundedness of N had not been made relative to L, condition 6 would also have entailed:

$$(\forall p \in P)\ (\forall \sigma \in \Sigma_{uc})\ F(\sigma,p) = 0 \tag{7}$$

Then, uncontrolable events would have been dealt with as unobservable events, which is a severe limitation. To sum up, searching for a controller net enforcing the "tolerance" [B_{min},B] on the behaviors of the plant amounts to constructing N satisfying conditions 3, 4, 5, and 6.

Let us consider have a glance at marked languages. When L_m is a marked language and $B = \overline{B} \cap L_m$, searching for a non-blocking controller net enforcing the tolerance [B_{min},B] amounts to constructing N satisfying conditions 3, 4, 5, and 6 (with $L = \overline{L_m}$).

It will be shown that the existence of such controller nets can be decided. *Here, the synthesis of controller nets does not rely at all upon Ramadge and Wonham's constructive results.*

C. Outline of the Decision Procedures

All problems we have met may be abstracted to a common form with three parameters: a prefix-closed regular language L, a regular subset $R \subseteq L$, and a regular language R'. Namely, they all reduce to finding among the L-bounded Petri nets some net N such that:

$$R \subseteq \mathcal{L}(N) \subseteq R' \cup C(L) \tag{8}$$

and N moreover conforms to the structural condition 3 (or 6 and 3). We shall construct a net N, conforming to these conditions, such that $R \subseteq \mathcal{L}(N)$ and $\mathcal{L}(N)$ is the *least* possible net language. It will then remain only to check the relation $\mathcal{L}(N) \cap L \subseteq R'$, which amounts to comparing two regular languages, because N is L-bounded.

IV. Petri Net Synthesis

We proceed in two steps. In the first step, we show how computing an *unbounded* net N such that $R \subseteq \mathcal{L}(N)$ and $\mathcal{L}(N)$ is the least possible. We observe that linear constraints on places of N (such as conditions 6 and 3) can be added without changing the algorithm. In the second step, L-boundedness is taken care of. For this purpose, we run the above algorithm iteratively for increasing sets of linear constraints until an L-bounded net is obtained. The number of iterations is at most one plus twice the size of the alphabet.

A. Unbounded Net Synthesis

From now on $N = (P,T,F,M_0)$, $T = \Sigma$, and R is a non empty prefix closed regular language of $\Sigma^* = T^*$ (this may be assumed w.l.o.g. for net languages are prefix closed).

Let $T = \{t_1,\ldots,t_n\}$. A place p of a net N may be represented as a $(2n+1)$-vector of non negative integers

$$< M_0(p), F(p,t_1),\ldots,F(p,t_n),F(t_1,p),\ldots,F(t_n,p) >$$

Assume $R \subseteq \mathcal{L}(N)$. Let $st_j \in R$ with $s \in \Sigma^*$ and let $[s]_i$ denote the occurrence count of t_i in s, then:

$$M_0 + \sum_{i=1}^{n} [s]_i \times (F(t_i,p) - F(p,t_i)) \geq F(p,t_j) \qquad (9)$$

Therefore, the set of all the places of all the nets N with languages larger than R coincides with the set of the vectors $\vec{x} =< x_0,x_1,\ldots,x_n,x_{n+1},\ldots,x_{2n} >$ such that for all $st_j \in R$:

$$x_0 + \sum_{i=1}^{n} [s]_i \times (x_{(n+i)} - x_i) \geq x_j \qquad (10)$$

When st_j ranges over $R \setminus \{\varepsilon\}$, we get an infinite set of linear inequality constraints on the variables x_0 $(= M_0(p))$, x_i $(= F(p,t_i))$ and $x_{(n+i)}$ $(= F(t_i,p))$. Using the fact that R is a regular language, this set may be reduced to a finite equivalent set of constraints. Two properties of the regular languages are crucial to the reduction. First, for any $j \in \{1,\ldots,n\}$, the quotient $R/t_j = \{s \in T^* \mid st_j \in R\}$ is regular. Second, for any regular language $S \subseteq T^*$, the set $[S] = \{[s] \mid s \in S\}$ (where $[s] =< [s]_1,\ldots,[s]_n >$) is a semi-linear subset of $\mathbb{N}^n$. Let us recall definitions. $\mathbb{N}^n$ is a monoid, with the componentwise addition of vectors as the product operation and the all-zeroes vector as the neutral element. A subset of $\mathbb{N}^n$ is *linear* if it may be expressed in this monoid as $\vec{e} \cdot \mathcal{F}^*$, where $\vec{e} \in \mathbb{N}^n$ and $\mathcal{F}$ is a finite subset of $\mathbb{N}^n$ ($\mathcal{F}^*$ is the least submonoid of $\mathbb{N}^n$ containing $\mathcal{F}$). A subset of $\mathbb{N}^n$ is *semi-linear* if it is a finite union of linear subsets.

For each $j \in \{1,\ldots,n\}$ and for each linear subset $\vec{e} \cdot \mathcal{F}^*$ of $[R/t_j]$, all instances of 10 that arise from words $s \in R/t_j$ such that $[s] \in \vec{e} \cdot \mathcal{F}^*$ may be replaced equivalently with the following *finite* system, where f ranges over $\mathcal{F}$:

$$\sum_{i=1}^{n} \vec{e}[i] \times (x_{(n+i)} - x_i) \quad \geq \quad x_j - x_0 \qquad (11)$$

$$\sum_{i=1}^{n} \vec{f}[i] \times (x_{(n+i)} - x_i) \quad \geq \quad 0 \qquad (12)$$

Therefore, the set of all places of all nets N with languages larger than R may be represented as the set of the non negative integer solutions $\vec{x} =< x_0,x_1,\ldots,x_n,x_{n+1},\ldots,x_{2n} >$ of a finite system of linear *homogeneous* inequalities 11 and 12. Let $\mathcal{S}_0$ denote this system. Chernikova's algorithm [8] allows to compute a finite family of non negative integer vectors $\vec{x}_1 \ldots \vec{x}_m$ such that the set of non negative solutions of $\mathcal{S}_0$ is the set of linear combinations $\vec{x} = \sum_{l=1}^{m} q_l \vec{x}_l$ with non-negative rational coefficients q_l.

This family of vectors $\vec{x}_1 \ldots \vec{x}_m$ induces a Petri net $N_0 = (P,T,F,M_0)$ with a set of places $P = \{p_1,\ldots,p_m\}$ as follows: $M_0(p_j)$ is the first component of $\vec{x}_j$, and for all i, $F(t_i,p_j)$ and $F(t_i,p_j)$ are the $(1+i)$-th and $(n+1+i)$-th components of $\vec{x}_j$, respectively. $\mathcal{L}(N_0)$ is actually the least net language larger than R. We refer the reader to [9] for a complete proof of this fact. The argument is as follows. Given any $s \in T^*$ and $t_j \in T$, if the linear inequality (9) does not hold for some place p of some net N such that $\mathcal{L}(N) \supseteq R$, then it does not hold for at least one place in the set $\{p_1,\ldots,p_m\}$, because the integer vector $\vec{x}$ representing p must be a solution of $\mathcal{S}_0$ and hence it is a non negative linear combination of $\vec{x}_1 \ldots \vec{x}_m$.

The above construction of N_0 stays unchanged when the condition 3 (or 6 and 3) is added as a constraint on controller nets N. It suffices indeed to augment the linear system $\mathcal{S}_0$ with one equation $x_i = x_{n+i}$ for each $t_i \in \Sigma_{uo}$ (and if necessary, with one equation $x_i = 0$ for each $t_i \in \Sigma_{uc}$). We assume from now on that these equations are comprised in $\mathcal{S}_0$.

B. L-bounded Net Synthesis

Let $A = (Q,\Sigma,\delta,q_0)$ be a finite deterministic automaton with the language $\mathcal{L}(A) = L$. Since L is prefix closed, all states $q \in Q$ are accepting states.

Let $N_0 = (P,T,F,M_0)$, where $T = \Sigma$, be the net with the least net language larger than R. It can be decided whether N_0 is L-bounded, in which case $\mathcal{L}(N_0)$ is clearly the infimum of the L-bounded net languages larger than R (the language Σ^* is always in this family since a net with no place is bounded).

The decision relays on the well known Karp and Miller's construction of a *covering tree* [10]. Starting from a root vertex v_0 labeled with (M_0,q_0), one constructs as many successor vertices as transitions $t \in T$ such that $M_0[t\rangle M$ for some M in N_0 and $\delta(q_0,t) = q$ for some $q \in Q$. Each new vertex v is labeled with the pair (M,q) determined by the corresponding t, and the incoming edge is labeled with t. This process is iterated from the new vertices, thus producing a tree. The expansion of the tree is stopped at each leaf vertex v with a label (M,q) greater than or equal to the label (M',q') of some ancestor vertex v', where $(M',q') \leq (M,q)$ *iff* $M' \leq M$ and $q' = q$. The sequence $s \in T^*$ read along the path from v' to v in the tree is then a repetitive sequence of $N_0 \times A$. Since Q and T are finite and by Dickson's lemma, the tree is finite. The net N_0 is L-bounded *iff* for each leaf vertex v, the label (M,q) of v is identical with the label of some ancestor vertex v'. In this case, the tree may be turned into a finite automaton that accepts the language $\mathcal{L}(N_0) \cap L$: it suffices to identify each leaf vertex v with the ancestor vertex v' that bears the same label.

If N_0 is not L-bounded, one may consider the set of the repetitive sequences s that increase the marking of N_0, i.e. that lead from some vertex v' to some leaf vertex v with a greater label. This set S is finite. For any sequence $s \in S$ from v' to v in the covering tree, let s' be the sequence read on the path from v_0 to v, then by construction of the tree, $s's^* \subseteq \mathcal{L}(N_0) \cap L$. Since $\mathcal{L}(N_0)$ is the least net language larger than R, a similar relation $s's^* \subseteq \mathcal{L}(N) \cap L$ holds for any Petri net N with language larger than R. Therefore, in order that N should be L-bounded, the following must hold for every place p of N:

$$\sum_{i=1}^{n} [s]_i \times (F(t_i,p) - F(p,t_i)) = 0 \tag{13}$$

Let $\mathcal{S}_1$ be the linear system formed of all equations and inequalities in $\mathcal{S}_0$, plus a new equation for each $s \in S$, as follows:

$$\sum_{i=1}^{n} [s]_i \times (x_{(n+i)} - x_i) = 0 \tag{14}$$

If some of the new equations was as a linear consequence of the constraints in $\mathcal{S}_0$, the corresponding sequence s would not increase the markings of N_0. Therefore, the new equations are independent of $\mathcal{S}_0$. Moreover, $\mathcal{S}_1$ is a finite linear system. After replacing $\mathcal{S}_0$ with $\mathcal{S}_1$, a new net N_1 may be computed following the algorithm described in IV-A. After replacing $\mathcal{S}_0$ and N_0 with $\mathcal{S}_1$ and N_1, respectively, one is back to the situation met at the beginning of IV-B, hence one can iterate the algorithm described in this section.

The iteration produces a sequence of pairs $(\mathcal{S}_k, N_k)$ for increasing k, starting with $k = 0$. We claim that it must stop at $k \leq 2n+1$ where n is the size of the alphabet. To show this, it suffices to recall that at each step, one adds at least one equation that does not depend on the equations already present. As places of nets are represented with vectors in $\mathbb{N}^{2n+1}$, this can occur at most $2n+1$ times, establishing the claim. In the particular case when the iteration stops at $k = 2n+1$, the resulting net has no place at all, since only the all-zeroes vector can be a solution of $\mathcal{S}_{2n+1}$.

V. Distributable Controller Nets

In this section, we add other constraints on Petri nets, reflecting distributed control architectures. We assume for this purpose that the alphabet $\Sigma = \Sigma_c \cup \Sigma_{uc} = \Sigma_o \cup \Sigma_{uo}$ comes equipped with a map $\lambda : \Sigma \to \{1, \ldots l\}$ assigning to each event the unique *location* where it may occur. This map induces a partition $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_l$ where $\Sigma_k = \lambda^{-1}(k)$. The goal is to implement Ramadge and Wonham's supervisory control with message passing automata $C_1 \ldots C_l$ such that each component automaton C_k controls (resp. observes) only the subset of events in $\Sigma_c \cap \Sigma_k$ (resp. in $\Sigma_o \cap \Sigma_k$), and the local controllers C_k act and communicate in fully asynchronous mode. Distributable Petri nets and their synthesis are a possible way to reach this goal.

A. Distributable Petri Nets

The short presentation below is inspired from [11]. We refer the reader to this paper for the detailed constructions and for the proofs.

Given a locating map from Σ to $\{1, \ldots l\}$, a *distributable* Petri net on Σ is a 5-tuple $N = (P, T, F, M_0, \lambda)$ where $T = \Sigma$, (P, T, F, M_0) is a Petri net, $\lambda : (P \cup T) \to \{1, \ldots l\}$ extends the locating map, and $F(p,t) \neq 0 \Rightarrow \lambda(p) = \lambda(t)$ for every place $p \in P$ and for every transition $t \in T$.

Any distributable Petri net $N = (P, T, F, M_0, \lambda)$ may be expanded to a Petri net $N' = (P', T', F', M'_0)$ as follows (see figure 4 for an illustration). Each place p of N is split to $l+1$ copies: a *local place* (p,k) for each location $k \in \{1, \ldots l\}$, and a *channel* $(p,0)$, thus $P' = P \times \{0, \ldots l\}$. For $t \in T$, let $F'((p,k),t) = F(p,t)$ if $\lambda(t) = k$, 0 otherwise. Similarly, let $F'(t,(p,k)) = F(t,p)$ if $\lambda(t) = k$, 0 otherwise. For each place p of N and for each location $k \neq \lambda(p)$, a *send* transition $k!p$ is added, such that $F'((p,k),k!p) = F'(k!p,(p,0)) = 1$ and $k!p$ is not connected to any other place. Finally, for each place p of N with the location $\lambda(p) = k$, a *receive* transition $k?p$ is added, such that $F'((p,0),k?p) = F'(k?p,(p,k)) = 1$ and $k?p$ is not connected to any other place.

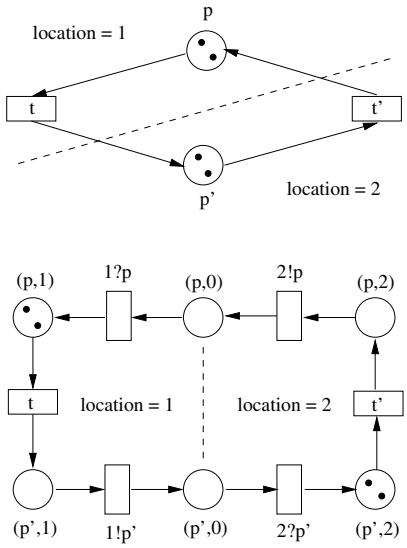

Fig. 4.

It has been proved in [11] that when send and receive transitions $k!p$ and $k?p$ are dealt with as silent transitions (hence labeled with ε), N' is divergence-free (every sequence of silent transitions is finite) and branching bisimilar to N. This means that there exists a binary relation $\sim$ between the reachable markings of N and N', respectively, such that:
$M_0 \sim M'_0$
$M_1 \sim M'_1 \wedge M'_1[*\rangle M'_2 \Rightarrow M_1 \sim M'_2$
$M_1 \sim M'_1 \wedge M'_1[t\rangle M'_2 \Rightarrow \exists M_2 \cdot M_1[t\rangle M_2 \wedge M_2 \sim M'_2$
$M_1 \sim M'_1 \wedge M_1[t\rangle M_2 \Rightarrow \exists M'_2 \cdot M'_1[*\rangle[t\rangle M'_2 \wedge M_2 \sim M'_2$
where $t \in T$ and $M'[*\rangle$ means the firing of some sequence of silent transitions. It was moreover shown that the relation $M \sim M'$ *iff* $\forall p \in P \; M(p) = \sum\{M'(p,k) \,|\, 0 \leq k \leq l\}$ satisfies all these conditions.

B. Distributed Implementations

We come back to supervisory control. Let $\Sigma = \Sigma_c \cup \Sigma_{uc} = \Sigma_o \cup \Sigma_{uo}$. Consider a plant where the events in Σ may occur at l different locations. Let $\lambda : \Sigma \to \{1,\dots l\}$ be the locating map, and let $\Sigma_k = \lambda^{-1}(k)$ for $k \in \{1,\dots,l\}$. Suppose this plant has a regular language L. Consider some supervisory control objective, and suppose that it may be enforced by a controller net N which is both L-bounded and distributable. The net N may be translated as follows to a system of message passing automata $C_1 \dots C_l$, where each automaton C_k controls (observes) the events in $\Sigma_c \cap \Sigma_k$ ($\Sigma_o \cap \Sigma_k$).

Since N is distributable, it may be expanded to a branching bisimilar net N' as defined in V-A. Let the product of N' and the plant be defined such that events in this product are either pairs $< t,t >$ where t is a transition of N or silent transitions $k!p$ or $k?p$. Since N is L-bounded, and the relation $M \sim M'$ *iff* $\forall p \in P \;\; M(p) = \sum\{M'(p,k) \,|\, 0 \le k \le l\}$ is a branching bisimulation between N and N', N' is L-bounded, and each place (p,k) inherits the bound of p in $N \times L$.

Removing all channels $(p,0)$ from N' yields net components $N'_1 \dots N'_l$ such that all places of N'_k have the form (p,k). For each $k \in \{1,\dots l\}$, an automaton C_k may be obtained by computing the reachability graph of N'_k within the bounds on places (p,k) inherited from the bounds of places p in $N \times L$. Thus, if N and N' are the nets from figure 4, C_1 is the automaton shown in figure 5, where the initial state is the marking $(2,0)$ (2 tokens in $(p,1)$ and no token in $(p',1)$).

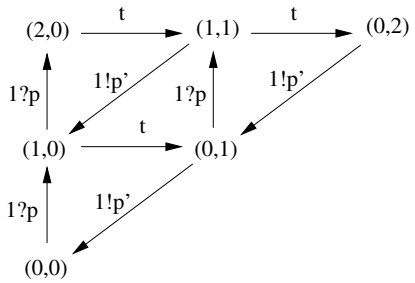

Fig. 5.

The message passing automata $C_1 \dots C_l$ send (receive) messages to (from) an asynchronous communication network which is only assumed not to loose any message. They yield the distributed implementation of the controller net N we were searching for. The naive algorithm we used to expand a distributable Petri net into a Petri net results in a high cost of communications. A more economic transformation, where each message contains more information, is defined in [11].

C. Distributable Petri Net Synthesis

It remains to show that for any instance of the problem 8, where L is a regular language over Σ and $\lambda : \Sigma \to \{1,\dots l\}$, one can construct an L-bounded and distributable net N such that 3 (or 6 and 3) hold, $R \subseteq \mathcal{L}(N)$, and these properties do not hold for any net N' unless $\mathcal{L}(N) \subseteq \mathcal{L}(N')$.

This can be done with a straightforward adaptation of the algorithm presented in IV-B. Every place p of a distributable net $N = (P,T,F,M_0)$, where $T = \Sigma$, must satisfy

$$(\exists k \in \{1,\dots,l\})\ (\forall t \notin \lambda^{-1}(k))\ F(p,t) = 0$$

Therefore, instead of considering the linear system $\mathcal{S}_0$ defined in IV-A, we consider now as many linear systems $\mathcal{S}_0^k$ as there are locations $k \in \{1,\dots,l\}$. Each linear system $\mathcal{S}_0^k$ is formed by augmenting $\mathcal{S}_0$ with one instance of the equation $x_j = 0$ for each $t_j \in T$ such that $\lambda(t_j) \neq k$.

For each $k \in \{1,\dots,l\}$, a net N_0^k may be constructed from $\mathcal{S}_0^k$ as indicated in IV-A. Let N_0 be the net which is obtained by gluing together all nets N_0^k on each transition $t \in T$. Clearly, $\mathcal{L}(N_0)$ is the least distributable net language larger than R. Given any automaton A with the language $\mathcal{L}(A) = L$, one may decide whether N_0 is L-bounded by constructing a covering tree for $N_0 \times A$. If N_0 is not L-bounded, one instance of the equation 14 is added to every system $\mathcal{S}_0^k$ for each repetitive sequence s which increases the markings of N_0. This yields a new family of linear systems $\mathcal{S}_1^k$. The iteration proceeds like in IV-A and it converges similarly in at most $2n+1$ steps, where n is the size of the alphabet.

As a final remark, it may be observed that any boolean combination of linear constraints $x_j = 0$ or $x_{n+j} = 0$ may be dealt with in a similar way, after is has been set to the disjunctive normal form: one constructs as many linear systems $\mathcal{S}_0^d$ as there are disjuncts d, formed by augmenting $\mathcal{S}_0$ with the constraints in the disjunct d.

Specific architectures of distributed controllers may therefore be taken into account within Petri net synthesis. For instance, if the communication network does not allow messages to be sent from location k to location h, one sets the constraint $\wedge_{\lambda(t_i)=k} \wedge_{\lambda(t_j)=h} (x_{n+i} = 0 \vee x_j = 0)$.

REFERENCES

[1] P.J. Ramadge and W.M. Wonham, Supervisory Control of a Class of Discrete Event Processes, *SIAM Journal of Control and Optimization*, vol. 25, 1987, pp 206-230.

[2] P.J. Ramadge and W.M. Wonham, On the Supremal Controllable Language of a Given Language, *SIAM Journal of Control and Optimization*, vol. 25, 1987, pp 637-659.

[3] P.J. Ramadge and W.M. Wonham, The Control of Discrete Event Systems, *Proc. of the IEEE, Special issue on Dynamics of Discrete Event Systems*, vol. 77, 1989, pp 81-98.

[4] A. Ghaffari and N. Rezg and X. Xie, Design of a Live and Maximally Permissive Petri Net Controller Using the Theory of Regions, *IEEE Transactions on Robotics and Automation*, vol. 19, 2003, 137-142.

[5] P.J. Ramadge and W.M. Wonham, "Modular Supervisory Control of Discrete Event Systems", *in Seventh Int. Conf. on Analysis and Optimization of Systems*, *LNCIS* vol. 83, Springer-Verlag, Germany, 1986, pp. 202-214.

[6] K. Rudie and S. Lafortune and F. Lin, Minimal Communication in a Distributed Discrete-Event System, *IEEE Transactions on Automatic Control*, vol. 48, 2003, pp. 957-975.

[7] K.G. Larsen, Modal Specifications, *in Automatic Verification Methods for Finite Transition Systems*, *LNCS* vol. 407, Springer-Verlag, New York, 1989, pp. 232-246.

[8] N. Chernikova, Algorithm for finding a general formula for the non-negative solutions of a system of linear inequalities, *USSR Computational Mathematics and Mathematical Physics* vol. 5, 1965, pp. 228-233.

[9] P. Darondeau, Unbounded Petri Net Synthesis, *in Lectures on Concurrency and Petri Nets*, Advances in Petri Nets, *LNCS* vol. 3098, Springer-Verlag, Berlin Heidelberg, 2004, pp. 413-438.

[10] R.M. Karp and R.E. Miller, Parallel Program Schemata, *Journal of Computer and System Sciences* vol. 3, 1969, pp. 147-195.

[11] E. Badouel and B. Caillaud and P. Darondeau, Distributing Finite Automata Through Petri Net Synthesis, *Formal Aspects of Computing* vol. 13, 2002, pp. 447-470.