Formal Aspects of Computing (2002) 13: 447–470
© 2002 BCS

Formal Aspects of Computing

Distributing Finite Automata Through Petri Net Synthesis

Éric Badouel[1], Benoît Caillaud[2] and P. Darondeau[2]
[1]ENSP, BP 8390, Yaoundé, Cameroun
[2]IRISA, Campus de Beaulieu, Rennes, France

Abstract. The synthesis problem for Petri nets consists in deciding constructively the existence of a Petri net with sequential state graph isomorphic to a given graph. If events are attached to locations, one may set as an additional requirement that the synthesised net should be distributable; i.e. such that events at different locations have no common input place, whence distributed conflicts are avoided. Distributable nets are easily implemented by finite families of automata (one per location) communicating with each other by asynchronous message passing. We show that the general Petri net synthesis problem and its distributed version may both be solved in time polynomial in the size of the given graph. We report on some preliminary experiments of Petri net synthesis applied to the distribution of reactive automata using the tool SYNET.

Keywords: Distribution; Finite automata; General Petri nets; Regions; Synthesis

1. Introduction

The *synthesis problem* for Petri nets is the question whether a given automaton (or a given language) on alphabet E is isomorphic to the state graph (or equal to the set of behaviours) of a Petri net to be discovered, with set of events identical to E. We will address this question for finite automata defined on an enriched alphabet (E, λ) where $\lambda : E \to \Lambda$ maps each event $e \in E$ to a *location* $\lambda(e) \in \Lambda$. Considering these automata as specifications of distributed systems, we search for realising them by *distributable* nets [Hop91], such that two transitions with different locations are never in conflict. A distributable net translates easily to an equivalent family of finite communicating automata which, plugged in at separate locations in an asynchronous network, provide the desired implementation. We give in this paper the description of an algorithm for distributed net synthesis and report on a few case studies where it supplies an assistance for the engineering of distributed protocols. Our goal is to attract engineers to a technique which may yield unexpected but effective solutions to practical distribution problems, relying in a totally hidden way on linear algebra that often beats intuition. A few case studies in distributed net synthesis would suffice to make the point, but we prefer to give a thorough presentation of the algorithm, yet unpublished, in order to allow users to try different implementations. The

Correspondence and offprint requests to: P. Darondeau, IRISA, Campus de Beaulieu, F-35042 Rennes Cedex, France. Email: `darondeau@irisa.fr` material of the paper is mainly assembled from the preliminary studies [BBD95] and [Cai99] and the research reports [BaD96] and [Cai97].

The problem of synthesising nets equivalent to a given finite automaton was addressed first by Ehrenfeucht and Rozenberg, who showed in [EhR90a] and [EhR90b] how to decide on the feasibility of this problem for elementary nets, using the crucial concept of *regions*. A (boolean) region in an automaton is a subset of states such that this set is entered or exited uniformly by all transitions with a common label (exactly as if it were the set of reachable cases of a net that hold a fixed condition or place). Regions may be interpreted as and give rise to *atomic* nets with a single place, filled by incoming transitions and emptied by outgoing transitions. By considering the finitely many different subsets of regions of a finite automaton, and gluing atomic nets on transitions, a finite number of nets may be derived in this way from a finite automaton, but it may well occur that none of them has a case graph isomorphic to the given automaton. Those automata which are isomorphic to reachable case graphs of elementary nets are characterised by two *separation axioms*, one expressing that any two different states are separated by some region containing exactly one of them, and the other expressing that for any event *e*, each state disabling *e* can be separated from all states enabling *e* by some region exited by *e*. When these axioms are satisfied, the automaton is isomorphic to the reachable case graph of the elementary net assembled from the whole set of regions (viewed as atomic subnets). The same holds of any smaller net assembled from a subset of regions large enough to witness the satisfaction of both separation axioms [DeR96].

The concept of regions was extended next by Mukund [Muk92], who defined regions modelling extensions of places in state graphs of general Petri nets with the step firing rule. Mukund's extended regions may again be interpreted as one-place Petri nets, with flow arcs now weighted by non-negative integers. Using these (integral) regions, Mukund established an abstract correspondence (namely a co-reflection) between separated *step* automata and Petri nets, where separated step automata satisfy an adapted version of Ehrenfeucht and Rozenberg's separation axioms. The separation axioms served also (albeit with slightly different regions) to characterise up to isomorphism automata which may be realised by pure Petri nets with the sequential firing rule [BDP96], and automata *with concurrency relation* which may be realised by general Petri nets [DrS93]. The aforementioned results do not directly entail the decidability of the Petri net realisation problem for finite automata, since the set of integral regions of a finite automaton is infinite: contrary to the set of boolean regions, this set cannot be inspected exhaustively in a finite amount of time for stating validity or invalidity of the separation axioms. It was shown in [BBD95] that the synthesis problem for *pure* Petri nets is nevertheless decidable, because the set of integral regions of a finite automaton forms a free module and one may actually compute a finite set of generators of this module. The decision was extended later on to general Petri nets, possibly accommodating *self-loops* [BaD96]. In both cases, deciding on the synthesis problem for Petri nets (or on the net realisation problem for finite automata) takes time polynomial in the size of automata.

In contrast, the synthesis problem for elementary nets is an NP-complete problem [BBD97]. The jump of complexity may be explained as follows. Integral regions, considered as maps from states to non-negative integers, are stable under addition of maps, hence they are suited for linear reasoning. On the contrary, regions, considered as maps from states to $\mathbb{Z}/2\mathbb{Z}$, are not stable under addition of maps, hence they are suited only for (non-linear) combinatorial reasoning. This did not prevent practically efficient algorithms based on binary decision diagrams and graph traversal to be constructed and used for deriving elementary nets from large automata [CKL95, CKL98]. The tool PETRIFY in which these algorithms are integrated shows convincing applications of net synthesis to asynchronous circuits [CKK96]. Elementary net synthesis finds there a privileged field of application, where it brings in a new technology.

The potential fields of application of general Petri net synthesis have not been investigated to a comparable degree. Following the path shown in [RXG00], applications may be found in supervisory control of discrete event systems, but the most critical issue of distributed and asynchronous control has not yet been considered in depth. More positively, the work presented in [Cai97, Cai99] brings the elements of an emerging technology for distributing finite reactive automata. Successful experiments have been conducted on simple communication protocols using the tool SYNET which has the nice feature of accommodating distribution constraints on the nets it is able to synthesise from automata.

The rest of the paper presents the principles of the synthesis of general Petri nets (Section 2), a proof that the synthesis problem can be solved in polynomial time for finite automata (Section 3), an adaptation of the synthesis algorithm to distributable nets (Section 4), a translation of distributable nets to asynchronously communicating automata (Section 5), two case studies in distributing reactive automata using SYNET (Section 6), and a few conclusions (Section 7).

2. The Petri Net Synthesis Problem

The purpose of the section is to give an axiomatic characterisation of the subfamily of finite automata which are isomorphic to reachable state graphs of unlabelled Petri nets. This characterisation is based on Ehrenfeucht and Rozenberg's separation axioms for elementary transition systems, adapted to general Petri nets through an adequate extension of the concept of regions. The extended regions we propose are similar in spirit to those considered by Mukund [Muk92] and by Droste and Shortt [DrS93, DrS96].

Before stating the definition of regions, let us fix terminology and notations. We assume a finite set E of *events*. A (finite) *automaton* over E is an initialised transition system $A = (S, E, T, s_0)$ with a (finite) non-empty set of *states* S, a (possibly empty) set of *transitions* $T \subseteq S \times E \times S$, and an *initial state* $s_0 \in S$.

For convenience, $s \xrightarrow{e} s'$ is an equivalent of $(s, e, s') \in T$, $\ell(s, e, s') = e$, and $s \xrightarrow{e}$ and $s \not\xrightarrow{e}$ are respective abbreviations for $\exists s' \in S\ (s \xrightarrow{e} s')$ and $\forall s' \in S\ \neg(s \xrightarrow{e} s')$. An automaton A is *deterministic* if for any event $e \in E$ and for any states $s, s', s'' \in S$, $(s \xrightarrow{e} s') \wedge (s \xrightarrow{e} s'') \Rightarrow s' = s''$. An automaton A is *co-deterministic* if for any state $s \in S$ and for any event $e \in E$, $(s' \xrightarrow{e} s) \wedge (s'' \xrightarrow{e} s) \Rightarrow s' = s''$. On the one hand, the automata we consider are not always deterministic or co-deterministic; on the other hand, they are always *reachable*, i.e. such that $S = \{s \mid s_0 \xrightarrow{*} s\}$, where $\xrightarrow{*}$ is the reflexive and inductive closure of the unlabelled transition relation $\rightarrow = \cup\{\xrightarrow{e} \mid e \in E\}$. An automaton is *event-reduced* if it is reachable and every event $e \in E$ occurs on at least one transition $(s, e, s') \in T$. We recall hereafter the definition of marked Petri nets and their sequential state graphs (see, e.g., [Rei85]).

Definition 2.1 (Petri nets). A (finite) Petri net is a triple $N = (P, E, F)$ where P and E are (finite) disjoint sets of *places* and *events*, and F is a function, $F : (P \times E) \cup (E \times P) \rightarrow \mathbb{N}$. The net is *pure* if $F(p, e) = 0$ or $F(e, p) = 0$ for every place p and for every event e; it is *impure* otherwise. A *marking* of N is a map $M : P \rightarrow \mathbb{N}$. An event e has *concession* at M if $M(p) \geqslant F(p, e)$ for every place $p \in P$. An event e which has concession at M may be *fired*, resulting in a transition $M\ [\ e > M'$ where M' is the marking such that $M'(p) = M(p) - F(p, e) + F(e, p)$ for every place $p \in P$.

Definition 2.2 (Marked nets and state graphs). A marked Petri net is a quadruple $\mathcal{N} = (P, E, F, M_0)$ where M_0 is a marking of the underlying net (P, E, F), called the *initial marking*. The marked net is *place simple* if it is never the case that $M_0(p) = M_0(p')$ for different places p and p' such that $[F(p, e) = F(p', e) \wedge F(e, p) = F(e, p')]$ for every event $e \in E$. The *reachable markings* of $\mathcal{N}$ are the markings $M : P \rightarrow \mathbb{N}$ such that $M_0\ [\ * > M$, where $[\ * >$ is the reflexive and transitive closure of the unlabelled transition relation of the underlying net (P, E, F). The sequential state graph of $\mathcal{N}$ is the automaton $\mathcal{N}^* = (RM(\mathcal{N}), E, T, M_0)$, where $RM(\mathcal{N})$ is the set of reachable markings of $\mathcal{N}$ and $T = \{M \xrightarrow{e} M' \mid M, M' \in RM(\mathcal{N}) \wedge M\ [\ e > M'\}$.

From this definition, the sequential state graph of a marked Petri net is a deterministic, co-deterministic and reachable automaton, but it is generally not finite. Nets considered in the sequel are place simple but not necessarily pure.

2.1. Regions

We come now to the definition of regions, which does not depend upon the finiteness of automata; still, we are mostly interested in regions of finite automata, for the axiomatic characterisation given in Section 2.2 is valid only for finite automata.

Definition 2.3 (Regions). A *region* of automaton $A = (S, E, T, s_0)$ is a tuple $(\sigma, {}^\bullet\eta\eta^\bullet)$ where $\sigma : S \rightarrow \mathbb{N}$, ${}^\bullet\eta$ and $\eta^\bullet : E \rightarrow \mathbb{N}$ are maps such that:

(a) $s \xrightarrow{e} \quad \Rightarrow \quad \sigma(s) \geqslant {}^\bullet\eta(e)$ and

(b) $s \xrightarrow{e} s' \quad \Rightarrow \quad \sigma(s') = \sigma(s) - {}^\bullet\eta(e) + \eta^\bullet(e)$

A region $(\sigma, {}^\bullet\eta, \eta^\bullet)$ is *pure* if ${}^\bullet\eta(e) = 0$ or $\eta^\bullet(e) = 0$ for every event $e \in E$.

It is readily observed that every place p of net $\mathcal{N} = (P, E, F, M_0)$ determines an associated region $(\sigma, {}^\bullet\eta, \eta^\bullet)$ of the sequential state graph $\mathcal{N}^*$, such that $\sigma(M) = M(p)$ for every reachable marking $M \in RM(\mathcal{N})$, and ${}^\bullet\eta(e) = F(p, e)$ and $\eta^\bullet(e) = F(e, p)$ for every event e.

Conversely, every region $p = (\sigma, {}^\bullet\eta, \eta^\bullet)$ of A determines an atomic net $\mathcal{N}' = (\{p\}, E, F', M_0')$, such that $F'(p,e) = {}^\bullet\eta(e)$, $F'(e,p) = \eta^\bullet(e)$, and $M_0'(p) = \sigma(s_0)$. In the case when $A = \mathcal{N}^*$ and region p derives from a homonymic place p of $\mathcal{N}$, the atomic net $\mathcal{N}'$ is actually isomorphic to the atomic subnet of $\mathcal{N}$ with the unique place p.

Before investigating the structure of the set of regions of an automaton, let us introduce notations and terminology based on an analogy with electricity specific to *pure regions.* In this particular case, the map $\eta = \eta^\bullet - {}^\bullet\eta$ provides the same information as the pair of maps $({}^\bullet\eta, \eta^\bullet)$. The map η measures the variations of σ along the paths of the automaton as a function of the events that occur, since

$$s \xrightarrow{e} s' \Rightarrow \sigma(s') - \sigma(s) = \eta(e) \tag{1}$$

on account of Condition *(b)* in Definition 2.3. Thus, if the automaton is reachable, the map σ is totally determined from $\sigma(s_0)$ and η; and the following condition is satisfied for every event $e \in E$:

$$[s_1 \xrightarrow{e} s_1' \quad \wedge \quad s_2 \xrightarrow{e} s_2'] \quad \Rightarrow \quad \sigma(s_1') - \sigma(s_1) = \sigma(s_2') - \sigma(s_2) \tag{2}$$

Given a region $(\sigma, {}^\bullet\eta, \eta^\bullet)$ of an automaton A, one may look at A as the model of an electric circuit, where each transition $s \xrightarrow{e} s'$ represents a component of type e, with nominal *tension* $\eta(e)$, plugged in between nodes s and s'. The map σ may thus be seen as a distribution of *potential.* Conversely, a map $\sigma : S \rightarrow \mathbb{N}$ satisfying Condition (2) defines a distribution of potential that can be realised by connecting components with adequate tensions $\eta(e)$, induced from differences of potential $\sigma(s') - \sigma(s)$ uniformly attached to transitions $s \xrightarrow{e} s'$. It will be assumed from now on that A is an event-reduced automaton, hence the map $\eta = \eta^\bullet - {}^\bullet\eta$ may be derived from σ for every region of A. For clarity, let us turn this into a formal definition.

Definition 2.4. Let $A = (S, E, T, s_0)$ be an event-reduced automaton. A map $\sigma : S \rightarrow \mathbb{N}$ satisfying Condition (2) is called a *distribution (of potential)* over the nodes of A. The *derived* map $\eta : E \rightarrow \mathbb{Z}$ such that σ and η satisfy together Condition (1) is called a *distribution (of tension)* over the events of A.

Extending the analogy further, we will show that if η derives from σ then $\sigma(s) = \sigma(s_0) + \int_{s_0}^{s} \eta$ for every state s (where integration follows any path from s_0 to s in A) and that $\int_C \eta = 0$ along every cycle C of A. For precision, let us state a few definitions.

Definition 2.5. The *underlying graph* of the automaton $A = (S, E, T, s_0)$ is the oriented multigraph $G(A) = (S, U, \partial^0, \partial^1)$ with components as follows:

- S, the set of *nodes*, is the set of states of A.
- U, the set of *arcs*, is the disjoint union of two copies of T, $U = T^+ \cup T^-$ where $\cdot^+$ and $\cdot^-$ are respective injections from T to U. Arcs $u \in T^+$ are *forward* transitions. Arcs $u \in T^-$ are *backward* transitions.
- The *source map* $\partial^0 : U \rightarrow S$ (and respectively the *target map* $\partial^1 : U \rightarrow S$) is defined with $\partial^0(t^+) = s$ and $\partial^0(t^-) = s'$ for $t = (s \xrightarrow{e} s') \in T$ (respectively with $\partial^1(t^+) = s'$ and $\partial^1(t^-) = s$).

Definition 2.6. A *path* in A or in $G(A)$ is a non-empty sequence of arcs $P = u_1 \dots u_n$ such that $\partial^1(u_i) = \partial^0(u_{i+1})$ for all $i < n$. The *source* and *target* of path P are the respective nodes $\partial^0(P) = \partial^0(u_1)$ and $\partial^1(P) = \partial^1(u_n)$. The *reverse* of path P is the path $P^{-1} = {u_n}^{-1} \dots {u_1}^{-1}$, where ${u_i}^{-1} = {t_i}^-$ if $u_i = {t_i}^+$ and ${u_i}^{-1} = {t_i}^+$ if $u_i = {t_i}^-$. Path P is *elementary* if $\partial^0(u_i) \neq \partial^0(u_j)$ and $\partial^1(u_i) \neq \partial^1(u_j)$ for all $i \neq j$. Path P is a *cycle* if $\partial^1(P) = \partial^0(P)$.

In the sequel, path and cycle will always be used to mean elementary path and elementary cycle.

Now for any map $\eta : E \rightarrow \mathbb{Z}$, let $\int_P \eta = \int_P^+ \eta - \int_P^- \eta$, where $\int_P^+ \eta$ is defined as the sum of $\eta \circ \ell(t_i)$ for forward transitions ${t_i}^+$ on path P and $\int_P^- \eta$ is defined as the sum of $\eta \circ \ell(t_i)$ for backward transitions ${t_i}^-$ on path P. It should thus be clear from conditions (1) and (2) that whenever $\sigma : S \rightarrow \mathbb{N}$ and $\eta : E \rightarrow \mathbb{Z}$ are compatible distributions of potential and of tension, the following identities do hold for every path P and for every cycle C:

$$\int_P \eta = \sigma(\partial^1(P)) - \sigma(\partial^0(P)) \tag{3}$$

$$\int_C \eta = 0 \tag{4}$$

This suggests the following alternative to Definition 2.4.

Definition 2.7. A map $\eta : E \to \mathbb{Z}$ satisfying equation (4) for every cycle C of the automaton A is a *distribution (of tension)* over the events of A.

A simple reasoning shows that the two definitions of distributions of tension given in Definitions 2.4 and 2.7 are equivalent for *finite* event-reduced automata. Suppose that $\int_C \eta = 0$ for every cycle C. It is easily seen that two paths P and P' such that both $\partial^0(P) = \partial^0(P')$ and $\partial^1(P) = \partial^1(P')$ can always be cut into finitely many slices P_i and P_i' such that $P_i = P_i'$ or $P_i(P_i')^{-1}$ is a cycle. It follows from the hypothesis on η that $\int_{P_i} \eta = \int_{P_i'} \eta$ for all i, whence $\int_P \eta = \int_{P'} \eta$. Therefore, there is nothing ambiguous if we let $\int_{s'}^{s} \eta$ be defined as $\int_P \eta$ for any path P from s' to s. It should now appear that η must derive from some distribution of potential: the adequate distributions $\sigma : S \to \mathbb{N}$ are obtained by integrating η according to

$$\sigma(s) = \sigma(s_0) + \int_{s_0}^{s} \eta \tag{5}$$

The resulting distributions σ are defined up to an additive constant, since any value of $\sigma(s_0)$ greater than or equal to the opposite of $\int_{s_0}^{s} \eta$ for all s (and in particular for s_0) may be chosen (recall that S has been assumed finite).

Now the opposite of $\int_{s_0}^{s'} \eta$ is $\int_{s'}^{s_0} \eta$ and $\int_{s'}^{s_0} \eta + \int_{s_0}^{s} \eta$ equals $\int_{s'}^{s} \eta$. We may therefore sum up as follows.

Proposition 2.8. The *pure* regions $(\sigma, {}^\bullet\eta, \eta^\bullet)$ of a finite event-reduced automaton A are in bijection with the pairs (η, k) such that $k \in \mathbb{N}$ and η satisfies equation (4) for every cycle C. The bijection is given by $\eta = \eta^\bullet - {}^\bullet\eta$ and $k = \mathbf{min}\{\sigma(s) \mid s \in S\}$. The reciprocal bijection is given by

$$\begin{array}{lcl} {}^\bullet\eta(e) & = & \mathbf{max}\{0, -\eta(e)\} \\ \eta^\bullet(e) & = & \mathbf{max}\{0, \eta(e)\} \\ \sigma(s) & = & \mathbf{max}\{\int_{s'}^{s} \eta \mid s' \in S\} + k \end{array}$$

We will now extend this characterisation from pure regions to *arbitrary regions* of A.

Proposition 2.9. The regions $(\sigma, {}^\bullet\eta, \eta^\bullet)$ of a finite event-reduced automaton A are in bijection with the triples (η, k, δ) such that $k \in \mathbb{N}$, η satisfies equation (4) for every cycle C, and $\delta : E \to \mathbb{N}$ is a map such that

$$\delta(e) \leqslant k + \mathbf{min}\{0, \eta(e)\} + \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \xrightarrow{e}\} - \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \in S\} \tag{6}$$

The bijection is given by $\eta = \eta^\bullet - {}^\bullet\eta$, $k = \mathbf{min}\{\sigma(s) \mid s \in S\}$, and $\delta(e) = {}^\bullet\eta(e)$ if $\eta(e) \geqslant 0$, $\delta(e) = \eta^\bullet(e)$ otherwise. The reciprocal bijection is given by

$$\begin{array}{lcl} {}^\bullet\eta(e) & = & \delta(e) + \mathbf{max}\{0, -\eta(e)\} \\ \eta^\bullet(e) & = & \delta(e) + \mathbf{max}\{0, \eta(e)\} \\ \sigma(s) & = & \mathbf{max}\{\int_{s'}^{s} \eta \mid s' \in S\} + k \end{array}$$

Proof. Let $(\sigma, {}^\bullet\eta, \eta^\bullet)$ be an arbitrary region of A. For every event $e \in E$, define:

$$\eta(e) = \eta^\bullet(e) - {}^\bullet\eta(e) \quad {}^\circ\eta(e) = \mathbf{max}\{0, -\eta(e)\} \quad \eta^\circ(e) = \mathbf{max}\{0, \eta(e)\}$$

From Definition 2.3, $(\sigma, {}^\circ\eta, \eta^\circ)$ is a pure region of A, hence there exists $k \in \mathbb{N}$ such that, for all $s \in S$, $\sigma(s) = \mathbf{max}\{\int_{s'}^{s} \eta \mid s' \in S\} + k$. Observe that $\eta(e) = \eta^\circ(e) - {}^\circ\eta(e)$. Therefore, $\eta^\bullet(e) - \eta^\circ(e) = {}^\bullet\eta(e) - {}^\circ\eta(e)$, and if $\delta(e)$ denotes this difference then $\delta(e) \geqslant 0$, for on the one hand, $\eta(e) \geqslant 0$ entails ${}^\circ\eta(e) = 0$ and $\delta(e) = {}^\bullet\eta(e) - {}^\circ\eta(e) = {}^\bullet\eta(e)$, and on the other hand, $\eta(e) < 0$ entails ${}^\circ\eta(e) = -\eta(e)$ and $\delta(e) = {}^\bullet\eta(e) + \eta(e) = {}^\bullet\eta(e) + (\eta^\bullet(e) - {}^\bullet\eta(e)) = \eta^\bullet(e)$. From condition (a) in Definition 2.3, ${}^\bullet\eta(e) \leqslant \sigma(s)$ whenever $s \xrightarrow{e}$, or yet equivalently ${}^\circ\eta(e) + \delta(e) \leqslant \mathbf{max}\{\int_{s'}^{s} \eta \mid s' \in S\} + k$. Now $\delta(e)$ satisfies relation (6), establishing half of the proposition, since

$$\mathbf{max}\{\int_{s'}^{s} \eta \mid s' \in S\} = \int_{s_0}^{s} \eta - \mathbf{min}\{\int_{s_0}^{s'} \eta \mid s' \in S\}$$

In order to establish the other half, consider η, k, and δ satisfying the conditions of the proposition, and let σ, ${}^\bullet\eta$ and $\eta^\bullet$ be the maps defined by the correspondence. We will show that $(\sigma, {}^\bullet\eta, \eta^\bullet)$ is a region of A. From Proposition 2.8, the map σ is a distribution of potential over the states of A, and η is the derived distribution of tension over the events of A. Since $\eta(e) = \eta^\bullet(e) - {}^\bullet\eta(e)$ for all e, condition (b) in Definition 2.3 is satisfied. Now, condition (a) in Definition 2.3 may be rewritten into the implication

$$s \xrightarrow{e} \quad \Rightarrow \quad \delta(e) - \mathbf{min}\{0, \eta(e)\} \leqslant k + \int_{s_0}^{s} \eta - \mathbf{min}\{\int_{s_0}^{s'} \eta \mid s' \in S\}$$

which follows directly from relation (6). □

The reader may observe that one shifts from pure regions to impure regions by adding simultaneously some $\delta(e)$ satisfying relation (6) to ${}^\bullet\eta(e)$ and to $\eta^\bullet(e)$ for each event e. Conversely, one returns from general regions to pure regions by subtracting simultaneously from ${}^\bullet\eta(e)$ and from $\eta^\bullet(e)$ the maximal $\delta(e)$ satisfying relation (6) for each event e.

Definition 2.10. Let $R_{\eta,k,\delta}$ denote the region of A fixed by the correspondence given in Proposition 2.9. In the case when $k = 0$ and δ takes for each e the maximal value allowed by relation (6), the region $R_{\eta,k,\delta}$ is said to be *canonical* and it is given the simpler notation R_η. Thus $R_\eta = (\sigma, {}^\bullet\eta, \eta^\bullet)$ is defined by

$$\begin{array}{rclcrcl} \sigma(s) & = & \sigma(s_0) + \int_{s_0}^{s} \eta & \text{where} & \sigma(s_0) & = & \mathbf{max}\{\int_{s}^{s_0} \eta \mid s \in S\} \\ {}^\bullet\eta(e) & = & \sigma(s_0) + \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \xrightarrow{e}\} & = & \mathbf{min}\{\sigma(s) \mid s \xrightarrow{e}\} \\ \eta^\bullet(e) & = & \eta(e) + {}^\bullet\eta(e) \end{array}$$

Observe that R_η may be computed from η using time polynomial in $|S|$ and $|E|$, which are both bounded by $|T| + 1$ (since A is reachable and event reduced). Canonical regions will play a major role in the sequel.

2.2. Representation Theorem

We come now to the logical laws explaining the structure of the sequential state graph of a net in terms of the regions that derive from its places.

Definition 2.11 (Separated automaton). An automaton $A = (S, E, T, s_0)$ is *separated* if and only if the following axioms hold for all states $s, s' \in S$ and for every event $e \in E$:

(**SSA**) $s \neq s' \Rightarrow \sigma(s) \neq \sigma(s')$ for some region $R = (\sigma, {}^\bullet\eta, \eta^\bullet)$
R solves the *states separation* problem at (s, s')

(**ESSA**) $s \overset{e}{\not\rightarrow} \Rightarrow \sigma(s) < {}^\bullet\eta(e)$ for some region $R = (\sigma, {}^\bullet\eta, \eta^\bullet)$
R solves the *event/state separation* problem at (s, e)

A subset of regions with enough elements to witness the satisfaction of both axioms is called an *admissible* subset of regions.

Let us explain the motivations under the definition of separated automata. If one observes the sequential state graph $\mathcal{N}^*$ of a marked Petri net $\mathcal{N}$, one may remark that it is a separated automaton: on the one hand, if markings M and M' are different, $M(p) \neq M'(p)$ for some place p, and the region that derives from p solves the states separation problem at (M, M'); on the other hand, if event e cannot be fired at M, $M(p) < F(p, e)$ for some place p, and the region that derives from p solves the event/state separation problem at (M, e). Thus, the regions of $\mathcal{N}^*$ that derive from the places of $\mathcal{N}$ form an admissible subset of regions, and axioms SSA and $ESSA$ are valid in every automaton isomorphic to the sequential state graph of a marked Petri net. Now reverting the analysis, consider a separated automaton A. Axiom SSA means that states of A may be represented injectively as markings of a net $\mathcal{N}$, whose places p are defined from regions $(\sigma, {}^\bullet\eta, \eta^\bullet)$ as one may expect: state s is mapped to marking M such that $M(p) = \sigma(s)$. In the considered representation, the contraposed version of axiom $ESSA$ means that an event e which has concession at marking M in $\mathcal{N}$ is necessarily enabled at s in A. As $\sigma(s)$ evolves through a transition $s \xrightarrow{e} s'$ in A in the same way as $M(p)$ evolves through a transition $M\ [\ e \rangle\, M'$ in $\mathcal{N}$, A and $\mathcal{N}^*$ cannot differ that much. We will show that the separation axioms characterise actually the variety of *finite event reduced* automata isomorphic to sequential state graphs of marked Petri nets. Recall that in an event-reduced automaton $A = (S, E, T, s_0)$, each event $e \in E$ labels at least one transition in T.

Lemma 2.12. A separated automaton is deterministic and co-deterministic.

Proof. Let $A = (S, E, T, s_0)$ be a separated automaton where $(s \xrightarrow{e} s')$ and $(s \xrightarrow{e} s'')$. Then for any region $(\sigma, {}^\bullet\eta, \eta^\bullet)$ of A, $\sigma(s') = \sigma(s) + \eta(e) = \sigma(s'')$ with $\eta = \eta^\bullet - {}^\bullet\eta$. Therefore, $s' = s''$ follows by axiom SSA. Co-determinism is shown in a similar way. □

Definition 2.13 (Net synthesised from finite subset of regions). Given an event reduced automaton $A = (S, E, T, s_0)$ and a finite subset P of regions of A, with typical region $p = (\sigma, {}^\bullet\eta, \eta^\bullet)$, let $A^*[P] = (P, E, F, M_0)$ be the marked Petri net such that $F(p, e) = {}^\bullet\eta(e)$, $F(e, p) = \eta^\bullet(e)$, and $M_0(p) = \sigma(s_0)$.

Theorem 2.14 (Characterisation result). A finite event reduced automaton is isomorphic to the sequential state graph of some marked Petri net if and only if it satisfies the separation axioms SSA and $ESSA$. A finite event reduced and separated automaton A is actually isomorphic to the sequential state graph of $A^*[P]$ for any finite admissible subset P of regions of A.

Proof. Let $A = (S, E, T, s_0)$ be a finite event reduced automaton. If this automaton is separated, it follows from finiteness that one can extract a finite admissible subset P from its infinite set of regions. It suffices to show that A is then isomorphic to the sequential state graph of $\mathcal{N} = A^*[P]$. Let $\mathcal{N} = (P, E, F, M_0)$, then by definition $\mathcal{N}^* = (RM(\mathcal{N}), E, T, M_0)$, where $RM(\mathcal{N})$ is the set of reachable markings of $\mathcal{N}$ and $\forall M, M' \in RM(\mathcal{N})$, $(M \xrightarrow{e} M') \in T$ if and only if $M\,[\,e > M'$ in $\mathcal{N}$. Define $\sim\, \subseteq (S \times RM(\mathcal{N}))$ such that $s \sim M$ if and only if for all regions $p \in P : p = (\sigma, {}^\bullet\eta, \eta^\bullet) \Rightarrow M(p) = \sigma(s)$. From this definition, $\sim$ is a functional relation. From the axiom SSA and the assumption that P is an admissible subset of regions of A, $\sim$ is an injective relation. We will show that $\sim$ is an isomorphism of automata. Observing on the one hand that $s_0 \sim M_0$, and on the other hand that both A and $\mathcal{N}^*$ are deterministic and reachable, it suffices to establish the following transfer property: if $s \sim M$ then for any $e \in E$, $s \xrightarrow{e}$ in A if and only if $M\,[e >$ in $\mathcal{N}$, and then $s' \sim M'$ where $s \xrightarrow{e} s'$ and $M\,[e > M'$. So let $s \sim M$. Suppose $s \xrightarrow{e}$ in A, then for every place $p \in P$: $p = (\sigma, {}^\bullet\eta, \eta^\bullet) \Rightarrow \sigma(s) \geqslant {}^\bullet\eta(e)$ by definition of regions and hence $M(p) \geqslant F(p, e)$, which shows that $M\,[e >$ in $\mathcal{N}$. Suppose $M\,[e >$ in $\mathcal{N}$ and assume for contradiction that $s \overset{e}{\not\to}$ in A. From the axiom $ESSA$ and the assumption that P is an admissible subset of regions of A, there exists in P some region $p = (\sigma, {}^\bullet\eta, \eta^\bullet)$ such that $\sigma(s) < {}^\bullet\eta(e)$, hence $M(p) < F(p, e)$ contradicting $M\,[e >$. Suppose finally that $s \xrightarrow{e} s'$ in A and $M\,[e > M'$ in $\mathcal{N}$. We should prove that for any place $p \in P : p = (\sigma, {}^\bullet\eta, \eta^\bullet) \Rightarrow M'(p) = \sigma(s')$. From the sequential firing rule, $M'(p) = M(p) - F(p, e) + F(e, p)$. From the definition of the net $\mathcal{N} = A^*[P]$, $F(p, e) = {}^\bullet\eta(e)$ and $F(e, p) = \eta^\bullet(e)$. From $s \sim M$, $M(p) = \sigma(s)$. From the definition of regions, $(s \xrightarrow{e} s') \Rightarrow \sigma(s) - {}^\bullet\eta(e) + \eta^\bullet(e) = \sigma(s')$. Altogether, $M'(p) = M(p) - F(p, e) + F(e, p) = \sigma(s) - {}^\bullet\eta(e) + \eta^\bullet(e) = \sigma(s')$, and $s' \sim M'$. □

Definition 2.15. Given an automaton A, a subset $\mathcal{R}$ of regions of A is *logically complete* if all instances in A of the separation problems solved by regions of A are equally solved by regions in $\mathcal{R}$.

Beware that logically complete sets of regions are generally not admissible. To explain this, consider for instance the automaton $A = (S, E, T, s_0)$ defined with $S = \{s_0, s_1\}$, $E = \{e\}$, and $T = \{s_0 \xrightarrow{e} s_1, s_1 \xrightarrow{e} s_0\}$. This automaton is not isomorphic to the sequential state graph of any marked Petri net: states s_0 and s_1 cannot be separated by any region of A. Thus the set of all regions of A is not admissible, but it is logically complete by trivial application of the definition. Another, more conclusive, example of a logically complete set of regions is, in any automaton, the subset of all regions $(\sigma, {}^\bullet\eta, \eta^\bullet)$ such that ${}^\bullet\eta(e) \neq 0$ and $\eta^\bullet(e) \neq 0$ may hold simultaneously for *at most one* event e: whenever two states s and s' are separated by a region, they are separated as well by a pure region; whenever a region solves the event/state separation problem at (s, e), the problem may be solved as well by a region such that ${}^\bullet\eta(e') = 0$ or $\eta^\bullet(e') = 0$ for every event $e' \neq e$. The point is that, for deciding on separation, it suffices to search for admissible subsets of some logically complete set of regions. This is the common principle of all net synthesis algorithms known so far, differing one from another on the choice of the logically complete set of regions. This set must be finite, or at least finitely generated, in order to support effective algorithms. The algorithm defined in the next section uses the fact that the set of *canonical* regions is logically complete.

3. A Polynomial Time Synthesis Algorithm

We describe in this section an algorithm for the synthesis of Petri nets from finite automata taking time polynomial in the size of the automata, thus establishing the following:

Theorem 3.1. Deciding whether a finite event-reduced automaton is isomorphic to the sequential state graph of some marked Petri net, and producing then a minimal Petri net realising the automaton, takes time polynomial in the number of transitions of the automaton.

The algorithm stems from the observation that the set of canonical regions is logically complete.

Proposition 3.2. In any finite event-reduced automaton A, the set of canonical regions R_η is logically complete, and a subset of canonical regions $\{R_\eta \mid \eta \in \mathscr{H}\}$ is admissible if and only if:

(i) $s \neq s' \quad \Rightarrow \quad \exists \eta \in \mathscr{H} \quad \int_{s_0}^{s} \eta \neq \int_{s_0}^{s'} \eta,$

(ii) $s' \overset{e}{\not\to} \quad \Rightarrow \quad \exists \eta \in \mathscr{H} \quad \int_{s_0}^{s'} \eta < \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \overset{e}{\to}\}.$

Proof. (i) By definition, a region $(\sigma, {}^\bullet\eta, \eta^\bullet)$ solves the states separation problem at (s, s') if and only if $\sigma(s) = \sigma(s_0) + \int_{s_0}^{s} \eta \neq \sigma(s_0) + \int_{s_0}^{s'} \eta = \sigma(s')$, where $\eta = \eta^\bullet - {}^\bullet\eta$. The separation of s from s' by the canonical region R_η is expressed by an identical condition, and this condition holds if and only if $\int_{s_0}^{s} \eta \neq \int_{s_0}^{s'} \eta$

(ii) By definition, a region $(\sigma, {}^\bullet\eta, \eta^\bullet)$ solves the event/state separation problem at (s', e) if and only if $\sigma(s') = \sigma(s_0) + \int_{s_0}^{s'} \eta < {}^\bullet\eta(e)$ where $\eta = \eta^\bullet - {}^\bullet\eta$. Now by Proposition 2.9, $(\sigma, {}^\bullet\eta, \eta^\bullet) = R_{\eta,k,\delta}$ for some $k \in \mathbb{N}$ and $\delta : E \to \mathbb{N}$ such that ${}^\bullet\eta(e) = \delta(e) + \mathbf{max}\{0, -\eta(e)\}$ and ${}^\bullet\eta(e) \leqslant k + \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \overset{e}{\to}\} - \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \in S\}$. As $\sigma(s_0) = k - \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \in S\}$ it comes that ${}^\bullet\eta(e) \leqslant \sigma(s_0) + \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \overset{e}{\to}\}$. To sum up, if the considered region $R_{\eta,k,\delta}$ separates s' from e, then necessarily $\int_{s_0}^{s'} \eta < \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \overset{e}{\to}\}$. By the definition of separation and Proposition 2.8, the canonical region R_η defined by $\eta = \eta^\bullet - {}^\bullet\eta$ separates s' from e if and only if $\sigma(s_0) + \int_{s_0}^{s'} \eta < \sigma(s_0) + \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \overset{e}{\to}\}$ and this holds if and only if $\int_{s_0}^{s'} \eta < \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \overset{e}{\to}\}$. □

The proof of Theorem 3.1 proceeds in two stages, described in separate subsections. We compute first a finite set of maps η generating all distributions of tension as linear combinations. We then use this finite presentation to check the separation conditions of Proposition 3.2. We are thus looking for an *admissible* family of distributions, containing enough elements to witness the satisfaction of both separation axioms. By Proposition 3.2, every admissible family of distributions $\mathscr{H}$ induces actually an admissible family of regions $\{R_\eta \mid \eta \in \mathscr{H}\}$ (where R_η is computed from η in polynomial time).

3.1. Computing Tensions

From now on, $A = (S, E, T, s_0)$ is a fixed automaton, finite and event-reduced.

Notation 3.3. Let $\mathbb{Z}[E]$ denote the set of maps $u : E \to \mathbb{Z}$ equipped with addition of maps and multiplication of maps by integers. Each map $u : E \to \mathbb{Z}$ may thus be written as a linear combination $\sum_{e \in E} u(e) \times \bar{e}$, where $\bar{e} : E \to \mathbb{Z}$ is the map $\bar{e}(e') = 1$ if $e = e'$, 0 otherwise. For instance, $u = a - 2b$ is the map $u(a) = 1$, $u(b) = -2$, and $u(e) = 0$ for $e \notin \{a, b\}$. Alternatively, a map $u : E \to \mathbb{Z}$ may be seen as an E-vector with entries $u(e) \in \mathbb{Z}$. The scalar product of u and v in $\mathbb{Z}[E]$ is the integer $u \cdot v = \sum_{e \in E} u(e) \cdot v(e)$.

Definition 3.4. The *Parikh image* of a path P (of the automaton A) is the map $\psi(P) : E \to \mathbb{Z}$ such that $\psi(P)(e) = \int_P \bar{e}$. The Parikh image of a cycle C is the map $\psi(C) : E \to \mathbb{Z}$ such that $\psi(C)(e) = \int_C \bar{e}$.

Example 3.5. The Parikh image of the cycle C going through states s_0, s_3, s_5, s_2, s_4, s_0 in the automaton of Fig. 1 is $\psi(C) = -a + a' + a + b' + a' = 2a' + b'$.

Proposition 3.6. A map $\eta \in \mathbb{Z}[E]$ defines a distribution of tension over the events of A if and only if $\eta \cdot \psi(C) = 0$ for every cycle C of A.

Proof. From Definition 2.7, η represents a distribution of tension over the events of A if and only if $\int_C \eta = 0$ for every cycle C. For each $e \in E$, define $\eta_e : E \to \mathbb{Z}$ such that $\eta_e(e') = \eta(e)$ if $e' = e$, 0 otherwise. Now $\int_C \eta = \sum_{e \in E} (\int_C \eta_e) = \sum_{e \in E} (\eta(e) \times \psi(C)(e)) = \eta \cdot \psi(C)$. □

Corollary 3.7. The maps $\eta \in \mathbb{Z}[E]$ that define distributions of tension over the events of A are the solutions of a linear system $M \cdot \eta = \mathbf{0}$, where M is an integral matrix whose rows are all Parikh images of (elementary) cycles of A.

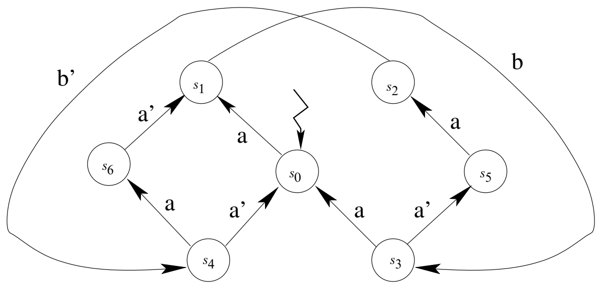


Fig. 1. An automaton.

As the number of (elementary) cycles of A is finite, the matrix M of the above corollary is well defined. At this stage, the classical algorithm of von zur Gathen and Sieveking (see [Sch86]) may be employed to compute (using time polynomial in the size of M) a finite set of solutions $\{\eta_1, \ldots, \eta_k\}$ of $M \cdot \eta = \mathbf{0}$ such that every solution of this system writes in a unique way as a linear combination $z_1\eta_1 + \ldots + z_k\eta_k$ with integer coefficients $z_i \in \mathbb{Z}$ (and any such combination yields a solution). Distributions of tension form therefore a *freely generated* integer module (see [MaB67] for a definition). Note that $k \leqslant |E|$ (since generators η_i are linearly independent) and hence $k \leqslant |T|$ (since $|E| \leqslant |T|$ by the hypothesis of event reducedness).

We have not paid much attention to algorithmic complexity in the above. As the number of (elementary) cycles of a finite automaton is not polynomial in the number of transitions, the situation is problematic: one cannot set any bound polynomial in $|T|$ on the number of rows of M (whose number of columns is bounded by $|T|$ as A is event-reduced). Fortunately, a standard result of applied graph theory (see e.g. [Che71] or [GoM79]) tells us that one needs not consider all cycles: it suffices to take as rows of M the Parikh images of $|T| - |S| + 1$ *fundamental* cycles of A (see Definition 3.8 below).

Let us recall briefly this result. An elementary cycle of A, $C = u_1 \ldots u_n$, may be represented by a map $C : T \to \{-1, 0, 1\}$ as follows: if $C = t^+t^-$ or $C = t^-t^+$ for some transition t, then let $C(t) = 0$ for all $t \in T$, else, for all $t \in T$, let $C(t) = +1$ if $u_i = t^+$ for some i, $C(t) = -1$ if $u_i = t^-$ for some i, and $C(t) = 0$ in any other case. The result is the following: every *spanning tree* of A (see Definition 3.8 below) determines a family of $\beta = |T| - |S| + 1$ cycles of A, let $\{C_1, \ldots, C_\beta\}$, such that any cycle of A writes as a linear combination $C = \sum_{j=1}^{\beta} x_j \times C_j$ with coefficients $x_j \in \{-1, 0, 1\}$. Thus, for any map $\eta : E \to \mathbb{Z}$:

$$\int_C \eta = \sum_{j=1}^{\beta} x_j \times \left(\int_{C_j} \eta \right)$$

As $\int_C \eta = \eta \cdot \psi(C)$ for every cycle C (see the proof of Proposition 3.6), it follows that η defines a distribution of tension over the events of A if and only if $M' \cdot \eta = \mathbf{0}$, where M' is the integral matrix whose rows are the Parikh images of the fundamental cycles $C_1 \ldots C_\beta$. Therefore, computing a set of generating distributions $\{\eta_1, \ldots, \eta_k\}$ takes time polynomial in the number of transitions $|T|$. It remains to give a precise definition of spanning trees and fundamental cycles, and to show an example.

Definition 3.8. Given a finite reachable automaton $A = (S, E, T, s_0)$, a *spanning tree* is a reachable automaton $B = (S, E, T', s_0)$ with an identical set of states, such that $T' \subseteq T$ and all elementary cycles of B have form $C = t^+t^-$ or $C = t^-t^+$ (hence B is a tree). The $|T| - |S| + 1$ transitions $t \in T \setminus T'$ are called *chords.* For each chord t, there exists a unique cycle C_t such that $C_t(t) = 1$ and $C_t(t') = 0$ for every chord $t' \neq t$ (thus $C_t(t') \neq 0 \;\Rightarrow\; t' = t \vee t' \in T'$). The *fundamental cycles* of A are the cycles C_t defined from the chords $t \in T \setminus T'$.

Note that *loops* $s \xrightarrow{e} s$ are special cycles, which appear as fundamental cycles under any choice of the spanning tree B. Note also that for each spanning tree B and for each state $s \neq s_0$ there exists a unique path from s_0 to s in B, denoted by P_s (hence, P_s is also a path from s_0 to s in A). Abusing the notations, let P_{s_0} denote the subgraph of A with the unique node s_0 and with the Parikh image $\psi(P_{s_0}) = \mathbf{0}$. Thus, for any state s and for any map $\eta \in \mathbb{Z}[E]$:

$$\int_{P_s} \eta = \sum_{e \in E} \left(\int_{P_s} \eta_e \right) = \sum_{e \in E} (\eta(e) \times \psi(P_s)(e)) = \eta \cdot \psi(P_s)$$

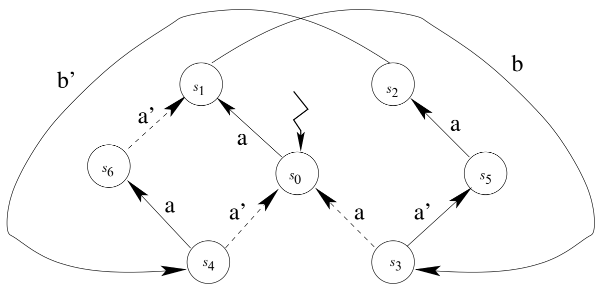


Fig. 2. An automaton with one of its spanning trees (in solid lines).

Table 1.

$\psi(P_{s_i})$	s_0	s_1	s_2	s_3	s_4	s_5	s_6
a	0	1	2	1	2	1	3
b	0	0	1	1	1	1	1
a'	0	0	1	0	1	1	1
b'	0	0	0	0	1	0	1

Example 3.9. The Parikh images of the fundamental cycles defined by the spanning tree indicated in solid lines in Fig. 2 are $2a + b$ and $2a + 2a' + b + b'$. Thus, $\eta \in \mathbb{Z}[E]$ is a distribution of tension if and only if $2\eta(a) + \eta(b) = 0$ and $2\eta(a') + \eta(b') = 0$. The distributions of tension are therefore generated by $\eta_1 = a - 2b$ and $\eta_2 = a' - 2b'$.

From now on, P_s is the path from s_0 to s in a fixed spanning tree, and $\{\eta_1, \ldots, \eta_k\}$ is a fixed set of generators for distributions of tension.

3.2. Solving the Separation Problems

Deciding upon separation of A comprises two subproblems, since two axioms must be satisfied. We consider first the states separation axiom SSA, which is the easier one to check. This axiom is valid in A if each pair of distinct states s and s' is *separated* by a distribution of tension η such that $\int_{s_0}^{s} \eta \neq \int_{s_0}^{s'} \eta$. As $\int_{s_0}^{s} \eta = \eta \cdot \psi(P_s)$ and thus depends linearly on η, s and s' are separated if and only if they are separated by some distribution in the generating set $\{\eta_1, \ldots, \eta_k\}$. In order to check this, it suffices to construct a table as follows: rows are indexed by generators $\eta_i \in \{\eta_1, \ldots, \eta_k\}$, columns are indexed by states $s \in S$, and the content of each entry (η_i, s) is the scalar product $\eta_i \cdot \psi(P_s)$. The axiom SSA is satisfied if and only if the columns of the table are all different (hence the distributions $\{\eta_1, \ldots, \eta_k\}$ form an admissible set w.r.t. state separation). In the converse case, the Petri net synthesis problem has no solution for A. As $|S| \leqslant |T| + 1$ (for all states are reachable), $k \leqslant |T|$, and there exist $|S| \times (|S| - 1)/2$ pairs of distinct states, checking states separation takes time polynomial in $|T|$.

Example 3.10. The Parikh images $\psi(P_s)$ of the paths from s_0 to s in the spanning tree of the automaton shown in Fig. 2 are shown in Table 1. The scalar products $\eta_j \cdot \psi(P_{s_i})$, where $\eta_1 = a - 2b$ and $\eta_2 = a' - 2b'$, are shown in Table 2. As all columns are different, the states separation axiom SSA is satisfied. Thus, the family of distributions $\{\eta_1, \eta_2\}$ is admissible w.r.t. state separation.

Table 2.

$\eta_j \cdot \psi(P_{s_i})$	s_0	s_1	s_2	s_3	s_4	s_5	s_6
η_1	0	1	0	-1	0	-1	1
η_2	0	0	1	0	-1	1	-1

Let us consider now the event/state separation axiom *ESSA*. This axiom is valid in A if each pair (s', e) made of a state s' and an event e disabled at s' is *separated* by a distribution (of tension) η such that

$$\int_{s_0}^{s'} \eta \ < \ \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \xrightarrow{e}\}$$

Define $\beta_i(s', s) = \eta_i \cdot \psi(P_{s'}) - \eta_i \cdot \psi(P_s)$ for $i \in \{1, \ldots, k\}$ and $s \in S$, and set $\eta = \sum_{i=1}^{k} z_i \times \eta_i$. The question is to decide whether one can find $z_i \in \mathbb{Z}$ solving the system of linear homogeneous inequalities

$$\left\{ \sum_{i=1}^{k} \beta_i(s', s) \times z_i < 0 \ \middle| \ s \xrightarrow{e} \right\} \tag{7}$$

The constants $\beta_i(s', s)$ may be computed in time polynomial in $|T|$. The number of the inequalities and the number of the unknown z_i are bounded by polynomials in $|T|$. Moreover, there are less than $|S| \times |E|$ instances of the event/state separation problem. Therefore, if (7) may be solved or proved unfeasible using time polynomial in the number of inequalities and in the number of the unknown then event/state separation may also be decided in polynomial time.

Now (7) is a *homogeneous* system of linear inequalities, hence it has an integral solution (in $\mathbb{Z}^k$) if and only if it has a rational solution (in $\mathbb{Q}^k$), and the integral solutions are the integer multiples of the rational solutions. Khachiyan's method of ellipsoids (see [Sch86]) may therefore be used to decide on the feasibility of (7) and to compute a solution, if it exists, in polynomial time.

By collecting the distributions $\eta = \sum_{i=1}^{k} z_i \times \eta_i$ that result from the solutions of (7) for all instances of *ESSA*, one obtains from $\{\eta_1, \ldots, \eta_k\}$ new distributions $\{\eta_{k+1}, \ldots, \eta_l\}$ which form an admissible set w.r.t. event/state separation.

Example 3.11. In our running example, the homogeneous system of linear inequalities that express event/state separation at (s_1, a) is the following:

s	$\sum_i \beta_i(s_1, s) \times z_i < 0$
s_0	$z_1 < 0$
s_3	$2z_1 < 0$
s_4	$z_1 + z_2 < 0$
s_5	$2z_1 - z_2 < 0$

This system, amounting to $z_1 < 0$ and $2z_1 < z_2 < -z_1$, is solvable. A solution is for instance $z_1 = -2$ and $z_2 = -3$. This solution determines a distribution of tension $\eta_3 = -2\eta_1 - 3\eta_2$ given by the map $\eta_3 = -2a + 4b - 3a' + 6b'$. The canonical region R_{η_3} derived from η_3 according to Definition 2.10 separates a from s_1 as desired. The maps $\{\eta_3, \eta_4, \eta_5\}$, where $\eta_4 = 2a - 4b + 3a' - 6b'$ and $\eta_5 = -a + 2b$, form an admissible set w.r.t. event/state separation.

Summary. The separation axioms may be checked within time polynomial in the number of transitions, yielding an admissible set of distributions $\{\eta_1, \ldots, \eta_l\}$. An admissible set of canonical regions $P = \{R\eta_1, \ldots, R\eta_l\}$ derives according to Definition 2.10. A is then isomorphic to the sequential state graph of $A^*[P]$ (Definition 2.13). A minimal admissible subset P' may be extracted from P using time polynomial in $|T|$, providing a minimal net realisation $A^*[P']$ of A. Theorem 3.1 is therefore proved.

Let us add one comment. When we call $A^*[P']$ a minimal realisation of A, we mean that no place can be suppressed while keeping a sequential state graph isomorphic to A; we do *not* mean that $|P'|$ is the minimal number of places needed for realising A.

Example 3.12. In the automaton of Fig. 2, all instances of the state separation problem, solved either by η_1 or by η_2 as we saw, are solved as well by η_3. The canonical regions $\{R_{\eta_3}, R_{\eta_4}, R_{\eta_5}\}$ which derive from $\eta_3 = -2a + 4b - 3a' + 6b'$, $\eta_4 = 2a - 4b + 3a' - 6b'$, and $\eta_5 = -a + 2b$ form therefore an admissible set of regions. Applying Definition 2.10 to produce canonical regions $R_\eta = (\sigma, {}^\bullet\eta, \eta^\bullet)$ from distributions of tension η, and using relations $F(p, e) = {}^\bullet\eta(e)$, $F(e, p) = \eta^\bullet(e)$, and $M_0(p) = \sigma(s_0)$ to derive places from regions R_η, one obtains the places p_3, p_4, and p_5 of the net displayed in Fig. 3. The sequential state graph of this net is actually isomorphic to the automaton of Fig. 1.

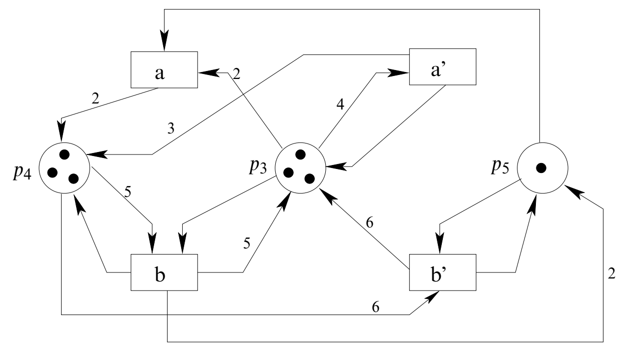


Fig. 3. net synthesised from canonical regions.

4. Adding Distribution Constraints

In order to show the principles of the application of net synthesis to the distribution of finite reactive automata, we introduce a special class of labelled Petri nets, called *distributable* nets. The label of an event indicates the location of its host in a network of automata which communicate by asynchronous message passing. The goal of distributable nets is to avoid distributed conflicts. This discipline is enforced by imposing common locations on conflicting events. The rest of the section deals with the synthesis of distributable nets from finite automata with fixed locations of events. This extended realisation problem can still be solved in polynomial time. An adapted synthesis algorithm has been implemented in SYNET.

Definition 4.1 (Distributable net system). A *distributable* net system with set of *locations* Λ is a quintuple $\mathcal{N} = (P, E, F, M_0, \lambda)$, where (P, E, F, M_0) is a marked Petri net and $\lambda : (P \cup E) \to \Lambda$ is a *placement* map such that $F(p, e) \neq 0 \Rightarrow \lambda(p) = \lambda(e)$ for every place $p \in P$ and for every event $e \in E$.

Our definition of distributable nets differs notably from Hopkins' definition given in [Hop91], since we impose from the start on placement maps a constraint strong enough to ensure the existence of distributed implementations. We postpone the discussion of distributed implementations to Section 5, and come now to the synthesis of distributable nets.

Definition 4.2 (Automata with set of locations). An automaton with set of *locations* Λ is a quintuple $A = (S, E, T, s_0, \kappa)$, where (S, E, T, s_0) is an automaton and $\kappa : E \to \Lambda$ is a *placement* map.

The synthesis problem for distributable nets (or the net realisation problem for automata with locations) consists in deciding from an automaton (S, E, T, s_0, κ) with set of locations Λ, given as input, whether the underlying automaton (S, E, T, s_0) is isomorphic to the sequential state graph of a distributable net (P, E, F, M_0, λ), to be constructed, with an identical set of locations Λ and such that λ extends κ. We shall produce a decision algorithm for this extended realisation problem by restricting the algorithm defined in Section 3 to a special class of regions, called *localisable* regions, such that conflict occurs exclusively between events with the same location in the induced atomic nets.

Definition 4.3 (Localisable regions). In an automaton with locations $A = (S, E, T, s_0, \kappa)$, a region $(\sigma, {}^\bullet\eta, \eta^\bullet)$ of (S, E, T, s_0) is *localisable* w.r.t. κ if $(\forall e', e'' \in E)\ {}^\bullet\eta(e') \neq 0 \wedge {}^\bullet\eta(e'') \neq 0 \Rightarrow \kappa(e') = \kappa(e'')$.

From Definitions 4.1 and 4.3, each place of a distributable net $\mathcal{N}$ with placement map $\lambda : (P \cup E) \to \Lambda$ determines a region of the sequential state graph $\mathcal{N}^*$ which is localisable with respect to $\lambda\lceil E$. A place p and the region $(\sigma, {}^\bullet\eta, \eta^\bullet)$ which it determines are in fact linked by the relation $F(p, e) = {}^\bullet\eta(e)$, hence ${}^\bullet\eta(e) \neq 0$ entails $\lambda(e) = \lambda(p)$, and ${}^\bullet\eta(e') \neq 0 \wedge {}^\bullet\eta(e'') \neq 0 \Rightarrow \lambda(e') = \lambda(e'')$. Therefore, the sequential state graph of a distributable net has always an admissible subset of localisable regions. Conversely, if P is an admissible subset of regions of the automaton $A = (S, E, T, s_0)$ and all regions in P are localisable with respect to $\kappa : E \to \Lambda$, the net $A^*[P]$ synthesised from P can be lifted to a distributable net. From Definition 4.3, one may actually extend κ to a placement map $\lambda : (P \cup E) \to \Lambda$ conform to Definition 4.1 by setting $\lambda(e) = \kappa(e)$ for $e \in E$, $\lambda(p) = \kappa(e)$ for places $p \in P$ such that $p = (\sigma, {}^\bullet\eta, \eta^\bullet)$ and ${}^\bullet\eta(e) \neq 0$ for some e, and by choosing arbitrarily $\lambda(p)$ for the remaining places $p \in P$ which do not fit this condition. To sum up:

Proposition 4.4. An automaton with locations $A = (S, E, T, s_0, \kappa)$ may be realised by a distributable net if and only if the underlying automaton (S, E, T, s_0) has an admissible subset P of localisable regions, in which case the net $A^*[P]$ synthesised from P may always be lifted to a distributable net.

Therefore, in order to decide on the net realisation problem for automata with locations, it suffices to decide on the restricted validity of the separation axioms with respect to localisable regions. Restricting validity of the separation axioms increases notably the complexity of the decision. It may therefore be wise, before deriving a distributable net from an automaton with locations, to derive first a Petri net from the underlying automaton. If this is not possible, the synthesis of the distributable net will certainly fail!

Henceforth, $A = (S, E, T, s_0)$ is a fixed automaton, finite, reachable and event reduced, $\kappa : E \to \Lambda$ is a surjective placement map with codomain $\Lambda = \{1, \ldots, m\}$, and $\{\eta_1, \ldots, \eta_k\}$ is a generating set of distributions of tension over E (see Section 3.1). We examine successively for the axioms SSA and $ESSA$ the conditions of their restricted validity with respect to the κ-localisable regions. The canonical regions which were used in Section 3 are not likely to fit in with distribution constraints. We shall consider here another logically complete set of regions, namely the regions of A which reach value 0 at some state $s \in S$ and that may therefore be called *strict* regions of A. These are all regions $R_{\eta,0,\delta}$ where η is a linear combination of the generators $\{\eta_1, \ldots, \eta_k\}$ and $\delta : E \to \mathbb{N}$ satisfies condition 6 of Proposition 2.9. As a result, we establish the following:

Theorem 4.5. Deciding whether a finite reachable and event reduced automaton with locations is isomorphic to the sequential state graph of a distributable net system and producing this net when it exists takes time polynomial in the number of transitions of the automaton.

4.1. Re-examining States Separation

Consider a pair of distinct states s' and s'' and suppose that $\sigma(s') \neq \sigma(s'')$ for some κ-localisable region $(\sigma, {}^\bullet\eta, \eta^\bullet)$. Let $\eta = \eta^\bullet - {}^\bullet\eta$. From $\sigma(s') \neq \sigma(s'')$ follows the assertion (i) that $\eta \cdot (\psi(P_{s'}) - \psi(P_{s''})) \neq 0$. From the assumption that $(\sigma, {}^\bullet\eta, \eta^\bullet)$ is κ-localisable, and since $\eta(e) + {}^\bullet\eta(e) \geqslant 0$ for all e, follows the assertion (ii) that there is at most one location $\iota \in \{1, \ldots, m\}$ such that $\eta(e) < 0$ for some event with location $\kappa(e) = \iota$. Conversely, if a distribution of tension η satisfies (i) and (ii), one may easily produce from η a κ-localisable region separating s' from s'': one may choose e.g. the region $R_{\eta,0,\mathbf{0}}$ (both strict and pure). Deciding on the existence of a κ-localisable region separating s' from s'' thus reduces to deciding whether (i) and (ii) are satisfied for some distribution of tension η.

Since η writes as a linear combination $\sum_i z_i \eta_i$ with coefficients $z_i \in \mathbb{Z}$, this can be done by solving or showing unfeasible each system in the indexed family $\Sigma_{\ltimes,\iota}$ of $2(m+1)$ homogeneous systems of linear inequalities in the z_i's defined as follows, with $\ltimes \in \{<, >\}$ and $\iota \in \{0, \ldots, m\}$. Each system $\Sigma_{\ltimes,\iota}$ has one strict inequality expressing condition (i), namely $\sum_i z_i (\eta_i \cdot (\psi(P_{s'}) - \psi(P_{s''}))) \ltimes 0$, plus inequalities enforcing (ii), namely one inequality $\sum_i z_i \cdot \eta_i(e) \geqslant 0$ for each event e such that $\kappa(e) \neq \iota$.

Because κ is a surjective map, m is bounded by $|E|$ and hence by $|T|$. As each system $\Sigma_{\ltimes,\iota}$ has size polynomial in $|T|$, the indexed family $\Sigma_{\ltimes,\iota}$ has size polynomial in $|T|$. Because all inequalities are homogeneous, $\Sigma_{\ltimes,\iota}$ has a solution in $\mathbb{Z}^k$ if and only if it has a solution in $\mathbb{Q}^k$. Thus each system in the indexed family $\Sigma_{\ltimes,\iota}$ may be solved up to a multiplicative factor or be shown unfeasible within time polynomial in $|T|$ following the ellipsoid method. Deciding whether the states separation axiom is valid with respect to κ-localisable regions and computing κ-localisable regions that witness its validity takes therefore time polynomial in $|T|$.

4.2. Re-examining Event/State Separation

Consider a state $s' \in S$ and an event $e' \in E$ disabled at s'. Suppose $\sigma(s') < {}^\bullet\eta(e')$ for some κ-localisable region $(\sigma, {}^\bullet\eta, \eta^\bullet)$. Then ${}^\bullet\eta(e') > 0$, hence ${}^\bullet\eta(e) = 0$ for every event e such that $\kappa(e) \neq \kappa(e')$. Therefore, if $\eta = \eta^\bullet - {}^\bullet\eta$, it holds (iii) that $\kappa(e) \neq \kappa(e') \Rightarrow \eta(e) \geqslant 0$ for all $e \in E$. Moreover, it holds (iv) that $\eta \cdot (\psi(P_s) - \psi(P_{s'})) > 0$ for every state s enabling e'. Conversely, if a distribution of tension η satisfies (iii) and (iv), one may always compute from η a κ-localisable region separating e' from s': one may choose e.g. the strict region $R_{\eta,0,\delta}$ with $\delta : E \to \mathbb{N}$ defined by

$$\delta(e') = \mathbf{m}in\{0, \eta(e')\} + \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \xrightarrow{e'}\} - \mathbf{min}\{\int_{s_0}^{s} \eta \mid s \in S\}$$

and $\delta(e) = 0$ for $e \neq e'$. Deciding upon the existence of a κ-localisable region separating e' from s' reduces therefore to deciding whether (iii) and (iv) are satisfied for some distribution of tension η.

Now letting $\eta = \sum_i z_i \cdot \eta_i$ with $z_i \in \mathbb{Z}$, consider the system of homogeneous linear inequalities in the unknown z_i's defined as follows: Σ has a first series of inequalities reflecting (iii), namely one inequality $\sum_i z_i \cdot \eta_i(e) \geqslant 0$ for each event e such that $\kappa(e) \neq \kappa(e')$, and a second series of inequalities reflecting (iv), namely one inequality $\sum_i z_i (\eta_i \cdot (\psi(P_s) - \psi(P_{s'}))) > 0$ for each state s enabling e'. All inequalities are homogeneous, hence Σ may be solved or shown unfeasible using time polynomial in $|T|$. Deciding whether the event/state separation axiom is valid with respect to κ-localisable regions and computing κ-localisable regions that witness its validity can therefore be done in polynomial time.

5. From Distributable Nets to Distributed Automata

5.1. Simple Distribution Scheme

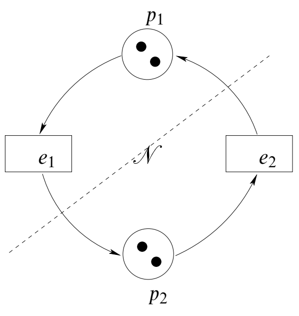
We have shown how to construct from a finite automaton with locations, when this is possible, a distributable net with the specified locations for events and with a sequential state graph isomorphic to the given automaton. This net is obviously *bounded*: each place p has a least upper bound $\overline{p}$ in the reachable markings. In order to complete the machinery for distributing finite automata, it remains to show that a bounded distributable net with m locations may always be implemented by m finite automata $A_1, \ldots, A_m$ communicating with each other by asynchronous message passing.

From now on, $\mathcal{N} = (P, E, F, M_0, \lambda)$ is a bounded distributable net with set of locations $\{1, \ldots, m\}$, hence $\lambda : (P \cup E) \to \{1, \ldots, m\}$. In order to produce a distributed implementation $\{A_1, \ldots, A_m\}$ of $\mathcal{N}$, we proceed in two stages. In the first stage, we extend $\mathcal{N}$ to a larger net $\mathcal{N}'$ in which (i) each place p of $\mathcal{N}$ is split to $m+1$ places: one *local place* (p,i) for each location $i \in \{1, \ldots, m\}$, and one *global place* $(p,0)$ whose tokens represent messages in transit, and (ii) *silent* events are supplied for *sending* tokens, i.e. for moving them from (p,i) to $(p,0)$ when $\lambda(p) \neq i$, or for *receiving* tokens, i.e. for moving them from $(p,0)$ to (p,i) when $\lambda(p) = i$. We show that $\mathcal{N}'$ is *divergence free* and *branching bisimilar* to $\mathcal{N}$ under the assumption that silent events are unobservable. In the second stage, we remove the global places. The effect of the removal is to disconnect $\mathcal{N}'$ and to produce m component nets $\mathcal{N}_i$. The finite automata A_i are obtained from the state graphs of the nets $\mathcal{N}_i$ by cutting out every marking in which some place (p,i) exceeds $\overline{p}$ (the least upper bound of p in $\mathcal{N}$).

We describe now the construction of $\mathcal{N}' = (P', E', F', M'_0)$. An illustration is given in Fig. 4 (where locations are indicated as subscripts). Each place of $\mathcal{N}$ is replicated $m+1$ times in $\mathcal{N}'$, thus $P' = P \times \{0, \ldots, m\}$. The initial marking is determined from the initial marking of $\mathcal{N}$ by the relation $M'_0((p,i)) = M_0(p)$ if $\lambda(p) = i$ and 0 otherwise, suggesting the privileged role of the place $(p, \lambda(p))$ among the representatives of p. E' is the union $E \cup S \cup R$ of the set of events of $\mathcal{N}$ and two new sets of events S (for sending) and R (for receiving). The flow of tokens attached to the events in E reproduces the flow of tokens in $\mathcal{N}$ up to the following adjustments. Let $F'((p,i),e) = F(p,e)$ if $\lambda(e) = i$ and 0 otherwise, and similarly let $F'(e,(p,i)) = F(e,p)$ if $\lambda(e) = i$ and 0 otherwise. As $\mathcal{N}$ is a distributable net, $F'((p,i),e) \neq 0 \Rightarrow \lambda(p) = i$, thus input and output are in fact dealt with asymmetrically. Let us come next to silent events. For each place p of $\mathcal{N}$ and for each location $i \neq \lambda(p)$, an event $i!p$ is supplied for sending tokens from the local place (p,i) to the global place $(p,0)$. Thus $S = \{\, i!p \mid 1 \leqslant i \leqslant m \wedge p \in P \wedge i \neq \lambda(p)\}$, and the flow of tokens attached to $i!p$ is defined by $F'((p,i), i!p) = F'(i!p, (p,0)) = 1$ and $F'(x, i!p) = F'(i!p, x) = 0$ for any other place $x \in P'$. Last, for each place p of $\mathcal{N}$ with location $\lambda(p) = i$, an event $i?p$ is supplied for receiving tokens into the local place $(p, \lambda(p))$ from the global place $(p,0)$. Thus $R = \{i?p \mid p \in P \wedge i = \lambda(p)\}$, and the flow of tokens attached to $i?p$ is defined by $F'((p,0), i?p) = F'(i?p, (p,i)) = 1$ and $F'(x, i?p) = F'(i?p, x) = 0$ for any other place $x \in P'$. This completes the definition of $\mathcal{N}'$.

We want to show that $\mathcal{N}$ and $\mathcal{N}'$ are equivalent up to an abstraction of the silent events $r \in R$ and $s \in S$. In order to define precisely this equivalence, let us relabel by some new symbol $\tau \notin E$ all the transitions labelled by $r \in R$ or $s \in S$ in the sequential state graph $\mathcal{N}'^{\,*} = (RM(\mathcal{N}'), E', T', M'_0)$. Next let $\sim \subseteq RM(\mathcal{N}) \times RM(\mathcal{N}')$ be the relation between the reachable markings of the respective nets $\mathcal{N}$ and $\mathcal{N}'$ such that $M \sim M'$ if and only if $M(p) = \sum\{M'((p,i)) \mid 0 \leqslant i \leqslant m\}$ for every $p \in P$. We shall prove the following facts:

1. $\mathcal{N}'^{*}$ is *divergence free*, which means that no infinite sequence of τ-labelled transitions takes place in this graph;
2. different markings of $\mathcal{N}$ have disjoint images under $\sim$, i.e., relation $\sim^{-1}$ acts functionally on $RM(\mathcal{N}')$;



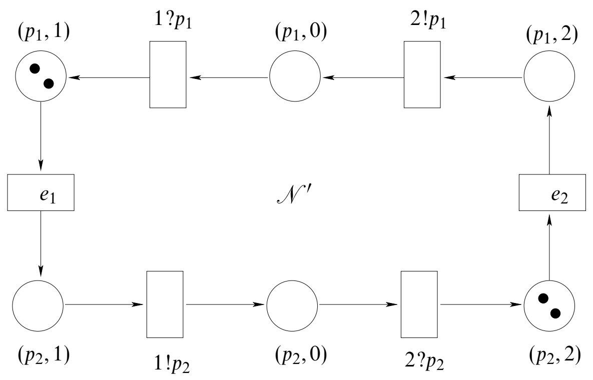


Fig. 4. Constructing $\mathcal{N}'$.

3. $\sim$ is a *branching bisimulation* [vGW89], which means in our specific case that the following assertions are valid, with e ranging over E, and with the subscripted M and M' ranging respectively over $RM(\mathcal{N})$ and $RM(\mathcal{N}')$.

   (a) $M_0 \sim M'_0$,

   (b) $M_1 \sim M'_1 \wedge M'_1 \xrightarrow{\tau} M'_2 \Rightarrow M_1 \sim M'_2$,

   (c) $M_1 \sim M'_1 \wedge M'_1 \xrightarrow{e} M'_2 \Rightarrow \exists M_2 \cdot M_1 \xrightarrow{e} M_2 \wedge M_2 \sim M'_2$,

   (d) $M_1 \sim M'_1 \wedge M_1 \xrightarrow{e} M_2 \Rightarrow \exists M'_2 \cdot M'_1 (\xrightarrow{\tau})^* \xrightarrow{e} M'_2 \wedge M_2 \sim M'_2$.

Relations **(a)** and **(b)** follow directly from the definitions of M', F' and $\sim$. The considered definitions also entail the following:

(e) $M_1 \sim M'_1 \wedge M_1 \xrightarrow{e} M_2 \wedge M'_1 \xrightarrow{e} M'_2 \Rightarrow M_2 \sim M'_2$,

(f) $M_1 \sim M'_1 \wedge M'_1 \xrightarrow{e} \Rightarrow M_1 \xrightarrow{e}$,

(g) $M_1 \sim M'_1 \wedge M_1 \xrightarrow{e} \wedge \forall p \cdot M_1(p) = M'_1(p, \lambda(p)) \Rightarrow M'_1 \xrightarrow{e}$.

Now $\mathbf{e} \wedge \mathbf{f} \Rightarrow \mathbf{c}$, and $\mathbf{b} \wedge \mathbf{e} \wedge \mathbf{g} \Rightarrow \mathbf{d}$ provided that in $\mathcal{N}'^*$ all maximal sequences of τ-labelled transitions originated from M'_1 lead to a marking M' such that $M'(p, \lambda(p)) = \sum\{M'_1((p,i)) \mid 0 \leqslant i \leqslant m\}$ for every $p \in P$.

For $p \in P$, let $\delta_p(M'_1) = \sum\{M'_1((p,i)) \mid 1 \leqslant i \leqslant m \wedge i \neq \lambda(p)\}$ and $\Delta_p(M'_1) = \sum\{M'_1((p,i)) \mid 0 \leqslant i \leqslant m \wedge i \neq \lambda(p)\}$. With these definitions, any maximal sequence of τ-labelled transitions originated from M'_1 includes exactly $\delta_p(M'_1)$ occurrences of sending events $i!p$ and $\Delta_p(M'_1)$ occurrences of receiving events $i?p$ for each $p \in P$. Thus, all maximal sequences of τ-labelled transitions originated from M'_1 have the same bounded length $\sum_p(\delta_p(M'_1) + \Delta_p(M'_1))$, which establishes fact 1. Moreover, by **(b)**, any maximal sequence of τ-labelled transitions originated from M'_1 reaches the indicated marking M', which establishes fact 2. Therefore, $\mathcal{N}$ and

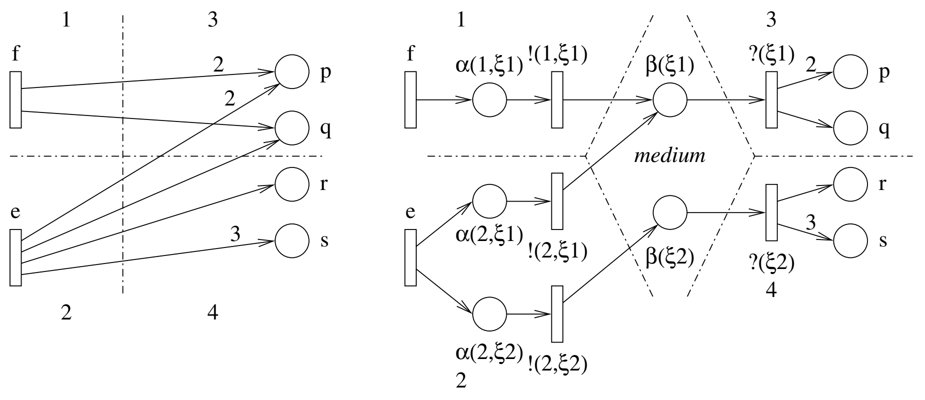


Fig. 5. Communications between locations.

$\mathscr{N}'$ behave in the same way up to an abstraction of the silent events; moreover, this simulation preserves distinctions between markings: separate markings of $\mathscr{N}$ have disjoint sets of images under relation $\sim$.

It remains to derive from $\mathscr{N}'$ finite communicating automata $A_1, \ldots, A_m$. For this purpose, we remove from $\mathscr{N}'$ the global places $(p, 0)$. What is left is a family of nets $\mathscr{N}'_i$, $i \in \{1, \ldots, m\}$, as follows: $\mathscr{N}'_i = (P'_i, E'_i, F'_i, M'_{i,0})$, P'_i is the set of local places with location i, E'_i is the set of events or silent events with location i, and F'_i and $M'_{i,0}$ are the induced restrictions of F' and M'_0. Thus $P'_i = P \times \{i\}$ and $E'_i = \{e \in E \mid i = \lambda(e)\} \cup \{\, i!p \mid p \in P \;\wedge\; i \neq \lambda(p)\} \cup \{\, i?p \mid p \in P \;\wedge\; i = \lambda(p)\}$. For each $i \in \{1, \ldots, m\}$, let A_i be the finite automaton that derives from the state graph of $\mathscr{N}'_i$ by cutting out all markings where some place (p, i) exceeds $\overline{p}$ (the least upper bound of p in $\mathscr{N}$). The resulting family $\{A_1, \ldots, A_m\}$ is a distributed implementation of $\mathscr{N}'$, and therefore of $\mathscr{N}$. Each automaton A_i is installed in the associated location i. An event $i!p$ (hence $i \neq \lambda(p)$) is interpreted by sending message p to the destination $\lambda(p)$. An event $i?p$ (hence $i = \lambda(p)$) is interpreted by looking for message p in the mailbox at the address $\lambda(p)$. Each message is supposed to reach its destination. No other assumption is made on synchronisation.

The markings of $\mathscr{N}'$ are thus represented by tuples $(s_0, s_1, \ldots, s_m)$, where $s_j : \{(p, j) \mid p \in P\} \to \mathbb{N}$: for $j = 0$, s_j represents the global state of the communication medium; for $j \in \{1, \ldots, m\}$, s_j is the local state of automaton A_j. This representation is injective; therefore, distinctions between markings are preserved by the distributed implementation of $\mathscr{N}$.

5.2. Optimised Distribution Scheme

We propose here two independent optimisations of the distribution scheme explained in Section 5.1, aiming to decrease the flow of messages on the communication medium (Section 5.2.1) or to decrease the size of the communicating automata (Section 5.2.2).

5.2.1. Aggregating Messages

The distribution scheme specified above leads to inefficient communication scenarios. For instance, two messages are needed for implementing a transition that produces tokens for two remote places even though they are mapped on the same location. A refined distribution scheme, producing automata with less communications, is presented now. It is based essentially on the same ideas, but fewer places are added and fewer tokens flow through the communication medium (see Fig. 5).

From now on, let $\mathscr{N} = (P, E, F, M_0, \lambda)$ be a distributable net system over set of locations Λ. Each event

$e \in E$ has an output flow $e^\bullet : P \to \mathbb{N}$ such that $e^\bullet(p) = F(e,p)$. This output flow may be decomposed thus into a *local* part $e^\bullet_{local} : P \to \mathbb{N}$ and a *remote* part $e^\bullet_{remote} : P \to \mathbb{N}$:

$$e^\bullet_{local}(p) = \left| \begin{array}{ll} e^\bullet(p) & \text{if } \lambda(e) = \lambda(p) \\ 0 & \text{otherwise} \end{array} \right.$$

$$e^\bullet_{remote}(p) = \left| \begin{array}{ll} e^\bullet(p) & \text{if } \lambda(e) \neq \lambda(p) \\ 0 & \text{otherwise} \end{array} \right.$$

The latter part may be further decomposed as $e^\bullet_{remote} = \Sigma_{l \in \Lambda}\, e^\bullet_l$ where $e^\bullet_l : P \to \mathbb{N}$ is the map

$$e^\bullet_l(p) = \left| \begin{array}{ll} e^\bullet(p) & \text{if } l = \lambda(p) \neq \lambda(e) \\ 0 & \text{otherwise} \end{array} \right.$$

For $e \in E$ and $l \in \Lambda$, define:

$$R_e = \{e^\bullet_l \,|\, l \in \Lambda,\ e^\bullet_l \neq \mathbf{0}\}, \quad R_l = \bigcup_{e \in \lambda^{-1}(l)} R_e \quad \text{and} \quad R = \bigcup_{l \in \Lambda} R_l$$

The distributable net system $\mathcal{N}' = (P', E', F', M'_0, \lambda')$ may be redefined as follows. A new location, representing the communication medium, is added: $\Lambda' = \Lambda \uplus \{medium\}$. Two types of communication places are added: on the one hand places $\alpha(l,\xi)$ used as output buffers, where each packet $\xi \in R_l$ is represented by a single token, and on the other hand places $\beta(\xi)$ used as communication buffers, where each token represents a packet in transit through the communication medium. This results in the set of places $P' = P \uplus \{\, \alpha(l,\xi) \,|\, l \in \Lambda,\ \xi \in R_l \,\} \uplus \{\, \beta(\xi) \,|\, \xi \in R \,\}$. Silent events $!(l,\xi)$ are supplied for moving packets $\xi \in R_l$ from output buffers $\alpha(l,\xi)$ to the communication buffer $\beta(\xi)$; silent events $?(\xi)$ are supplied for picking packets ξ from the communication medium and dispatching them on arrival. This results in the set of events $E' = E \uplus \{\, !(l,\xi) \,|\, l \in \Lambda,\ \xi \in R_l \,\} \uplus \{\, ?(\xi) \,|\, \xi \in R \,\}$. Let the locations of the new places and events be as follows:

$$\begin{array}{lcl} \lambda'(\alpha(l,\xi)) & = & l \\ \lambda'(\beta(\xi)) & = & medium \\ \lambda'(!(l,\xi)) & = & l \\ \lambda'(?(e^\bullet_l)) & = & l \end{array}$$

The other locations are like in $\mathcal{N}$. For places $p \in P$ and events $e \in E$ inherited from $\mathcal{N}$, the flow relation F' is defined thus: $F'(x,y) = F(x,y)$ if $\lambda(x) = \lambda(y)$, 0 otherwise. The flow relation F' is extended as follows to the new places or events, with $F'(x,y) = 0$ in all cases left unspecified:

$$\begin{array}{ll} (\, e \in E,\ \xi \in R_e \,) & F'(e, \alpha(\lambda(e),\xi)) = 1 \\ (\, e \in E,\ \xi \in R_e \,) & F'(\alpha(\lambda(e),\xi), !(\lambda(e),\xi)) = 1 \\ (\, l \in \Lambda,\ \xi \in R_l \,) & F'(!(l,\xi), \beta(\xi)) = 1 \\ (\, \xi \in R \,) & F'(\beta(\xi), ?(\xi)) = 1 \\ (\, \xi \in R,\ p \in P \,) & F'(?(\xi), p) = \xi(p) \end{array}$$

One proves that $\mathcal{N}'^*$ is divergence free and branching bisimilar to $\mathcal{N}^*$ in the same way as in Section 6.1, but using now the alternative relation $M \sim M'$ if and only if $\forall p \in P$, $M(p) = M'(p) + \Sigma_{\xi \in R}\ \xi(p) \times M'(\beta(\xi)) + \Sigma_{i \in \Lambda, \xi \in R}\ \xi(p) \times M'(\alpha(i,\xi))$.

A distributed implementation of $\mathcal{N}'$ follows along the same lines as in Section 5.1: the automata A_i are obtained by expanding the state graphs of the sub-nets $\mathcal{N}'_i$ of $\mathcal{N}'$, with bounds $\overline{p}$ set on places $(p,i) \in P'_i$.

5.2.2. *Pruning Automata*

A drawback of the distribution schemes described so far is to produce automata bigger than needed: the automata A_i may be ready to accept inputs from the communication medium at states where no input will ever come! One may remedy this drawback by computing directly the A_i from the state graph of $\mathcal{N}'$. The stages of the computation are as follows. For each $i \in \{1, \ldots, m\}$, let A'_i be the automaton obtained from $\mathcal{N}'^*$ by renaming with τ ($\notin E'$) all transitions $M \xrightarrow{e} M'$ such that $e \in E' \setminus E'_i$ (E'_i is defined as in Section 5.1). Next let A_i'' be the finite non-deterministic automaton whose transitions $M \xrightarrow{e} M'$ are derived from the sequences of transitions $M = M_1 \xrightarrow{\tau} M_2 \dots \xrightarrow{\tau} M_n \xrightarrow{e} M'$ in A_i' such that $e \neq \tau$. Finally let A_i be the finite deterministic automaton which derives from A_i'' according to the traditional subset construction. Since the subset of events $E' \setminus E_i'$ thus abstracted from is precisely the subset of events which do not affect the places in P_i', the automaton A_i is isomorphic to the induced restriction of $\mathscr{N}'^*_i$ on the subset of markings of $\mathscr{N}'_i$ which occur as sub-markings of reachable markings of $\mathscr{N}'$. Therefore, the system of communicating automata $\{A_1, \dots, A_m\}$ has a state graph isomorphic to $\mathscr{N}'^*$. Since the bisimulation $\sim^{-1}$ between $\mathscr{N}'^*$ and $\mathscr{N}^*$ acts functionally on $RM(\mathscr{N}')$, $\{A_1, \dots, A_m\}$ is a distributed implementation of $\mathscr{N}$, and this implementation preserves distinctions between markings.

Remark 5.1. If one does not insist on realising $\mathscr{N}'^*$ up to isomorphism, one can go one step further by minimising the automata A_i. The language of $\mathscr{N}$ then coincides with the set of sequences of events in E performed by the communicating automata.

Remark 5.2. In order to abstract from a subset of events $\mathscr{E} \subseteq E$, it suffices to redefine A_i' as the copy of $\mathscr{N}'^*$ in which all events in $(E' \setminus E_i') \cup \mathscr{E}$ have been replaced with τ. If $\mathscr{N} = A^*[P]$, the set of sequences of events in $E \setminus \mathscr{E}$ that may be performed by the communicating automata A_i coincides now with the language of the $\mathscr{E}$-collapse of A (the automaton obtained by collapsing every pair of states s and s' such that $s \xrightarrow{e} s'$ and $e \in \mathscr{E}$).

6. Case Studies in Distributing Reactive Automata

This section details two applications of distributable Petri net synthesis. The first case is the systematic derivation of two distributed protocols for mutual exclusion. The second case is the synthesis of a simplified transport level communication protocol derived from the INRES protocol [Hog91]. Both examples illustrate a new methodology for distributed program synthesis, based on the algorithms for distributable Petri net synthesis presented in the previous sections: a distributable Petri net is first synthesised and then turned into a collection of communicating finite state automata, each of which defines the behaviour of the sequential process located at a specific site of the distributed architecture. We could have contented ourselves with distributing the synthesised Petri net over the distributed architecture without going to sequential machines. This would fit nicely if the goal was to implement protocols in hardware, since concurrency at each site could then be exploited within circuit synthesis. However, this would not meet the customary requirements for software implementations of low-level protocols : each site is one processor, and emulating concurrency would be too inefficient.

6.1. Mutual Exclusion

6.1.1. Introduction

Mutual exclusion is one of the basic services that are often present in distributed operating systems. We focus on the possible ways of achieving mutual exclusion on an asynchronous network of processors in which communication is performed by message passing and such that no message shall be lost or replicated. The specification of mutual exclusion between two users U_1 and U_2 is given in Fig. 6. Event r_i, $i \in \{1,2\}$ corresponds to a request by user U_i to enter critical section. The meaning of event e_i, $i \in \{1,2\}$ is that user U_i is allowed to enter critical section while event x_i, $i \in \{1,2\}$ corresponds to the exit of user U_i from critical section. Mutual exclusion is ensured whenever both agents cannot be in critical section at the same time.

A distributed implementation of mutual exclusion consists in a pair of sequential processes P_1, P_2 where process P_i, $i \in \{1,2\}$ controls user U_i and both processes communicate only by asynchronous message passing. We wish to apply the method sketched in Sections 4 and 5 to the derivation of processes P_1 and P_2. Unfortunately, the automaton in Fig. 6 is not isomorphic to the reachability graph of any distributable Petri net. This is due to the conflict between events e_1 and e_2 in state 4 while these two events are mapped to distinct locations.

The next sections describe two ways of inserting silent events in the automaton of Fig. 6 so that distributable Petri net synthesis becomes feasible. The process of inserting silent transitions is manual and

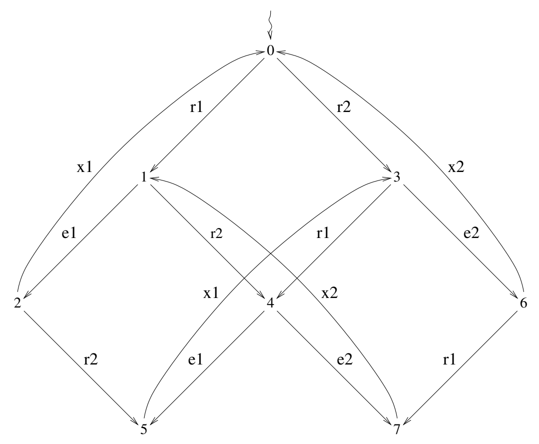


Fig. 6. Specification of the mutual exclusion service.

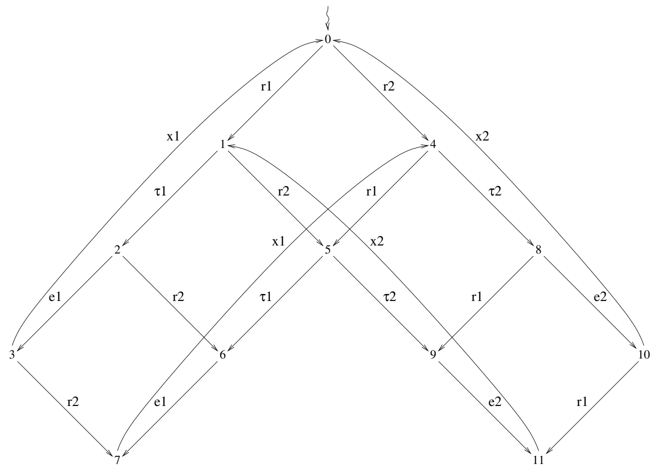


Fig. 7. Mutual exclusion: a first refinement.

does rely solely on designer's intuition. In this respect, it doesn't differ in any way from silent transition insertion in [Yak98].

6.1.2. A Monitor-Based Solution

A first solution consists in refining event e_i, $i \in \{1, 2\}$ by $\tau_i e_i$, where τ_i is located on a third process M. This leads to the automaton in Fig. 7 which is comformant to the original automaton with respect to the **ioco** testing equivalence [Tre96]. The refined automaton is isomorphic to the reachability graph of the distributable Petri net in Fig. 8.

Communicating automata can then be produced from the distributable Petri net, following the method described in Sections 5.1 and 5.2.2. By abstracting from the subset of events $\mathscr{E} = \{\tau_1, \tau_2\}$, and by applying minimisation, one obtains the automata shown in Fig. 9. The process at location M acts as a monitor for the two other processes $P_i, i = 1 \dots 2$, which in turn act as interfaces between user processes U_i and the monitor M. It follows from the method that the implementation is behaviourally correct, hence all and only the sequences of events which are compatible with the service specifications in Fig. 6 can occur. In spite of the minimisation, the automaton in Fig. 6 can moreover be reconstructed from the distributed implementation.

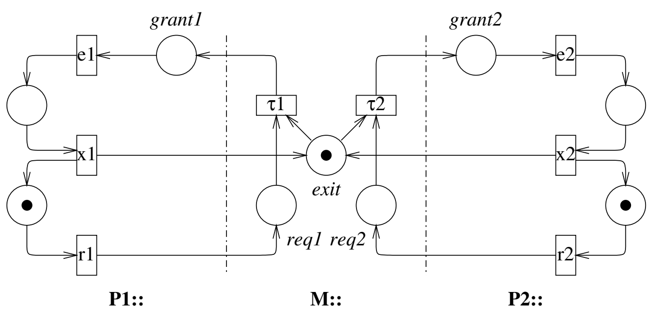


Fig. 8. Mutual exclusion: distributable Petri net generated from the first refinement.

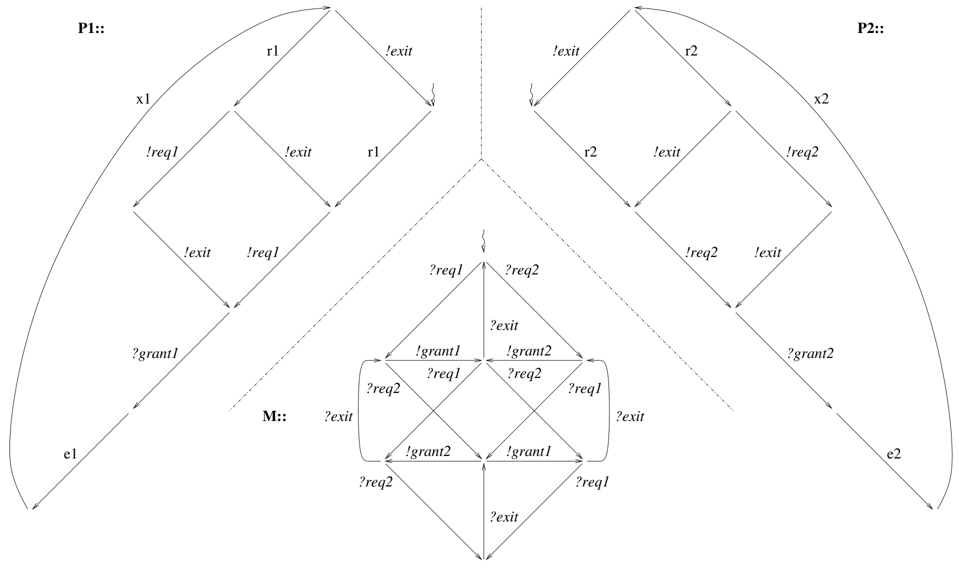


Fig. 9. Mutual exclusion: communicating automata generated from the first refinement.

6.1.3. *A Token-Based Solution*

A second solution to the mutual exclusion protocol synthesis problem is to split states 0, 1, 3 and 4 and to insert converse transitions labelled by silent events τ_1 and τ_2 as represented in Fig. 10. Event τ_i, $i \in \{1, 2\}$ is located on process P_i; therefore no distributed conflict is created and, more importantly, the distributed conflict between events e_1 and e_2 is alleviated. Not surprisingly, the reachability graph of the synthesised Petri net shown in Fig. 11 is isomorphic to the refined automaton.

Two communicating automata may be derived from the distributable Petri net, following the method described in Sections 5.1 and 5.2.2. By abstracting from the subset of events $\mathscr{E} = \{\tau_1, \tau_2\}$, one obtains the automata shown in Fig. 12; mutual exclusion is thus achieved by circulating a token between processes P_1 and P_2. Minimisation has not been used: the automata produced by determinisation are already minimal. It follows therefore from the method that the automaton in Fig. 6 can be reconstructed from this distributed implementation.

6.2. A Simplified INRES Protocol

In this section, we consider a simplified version of the INRES communication protocol [Hog91]. The specifications of service of the INRES protocol are given in Fig. 13. For the sake of the exposition, we consider the simplified service described in Fig. 14. The simplified protocol defines the transmission of data between two users *user A* and *user B*, linked to respective protocol entities A and B. Event s means that entity

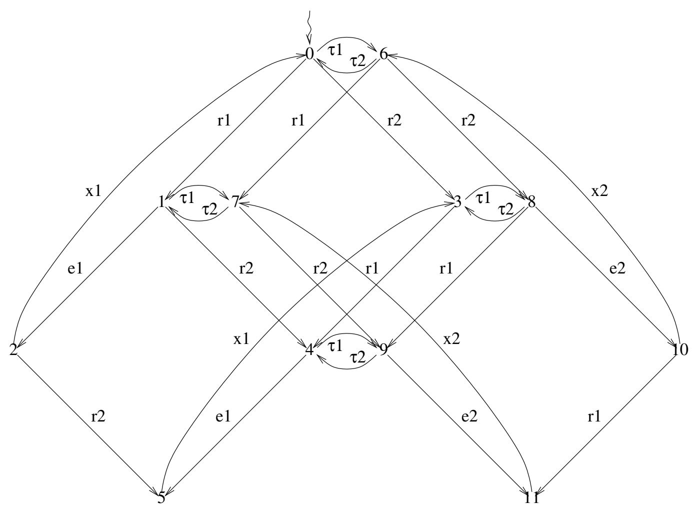


Fig. 10. Mutual exclusion: a second refinement.

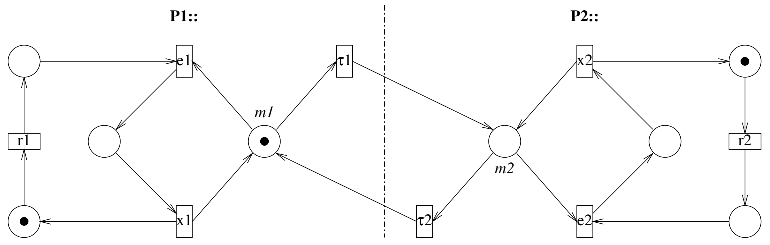


Fig. 11. Mutual exclusion: distributable Petri net generated from the second refinement.

A is given by *user A* some data to transmit; event r means that entity B delivers some data to *user B*; event d is a disconnection request (located on B); event a is the notification of the disconnection (located on A). With this protocol, data exchanges (words in $(sr)^\star$) may take place until a disconnection is requested.

We aim at synthesising a distributed implementation of this protocol: two processes A and B, communicating with one another through a reliable communication medium. Events s and a are located on process A, while events r and d are located on B. This is defined by the location map λ: $\lambda(s) = \lambda(a) = A$ and $\lambda(r) = \lambda(d) = B$. We wish to use *distributable* Petri net synthesis to produce a Petri net implementation of the communication protocol: thus, for every place p of the net, the flow relation F must satisfy:

$F(p, s) = F(p, a) = 0$ or $F(p, r) = F(p, d) = 0$

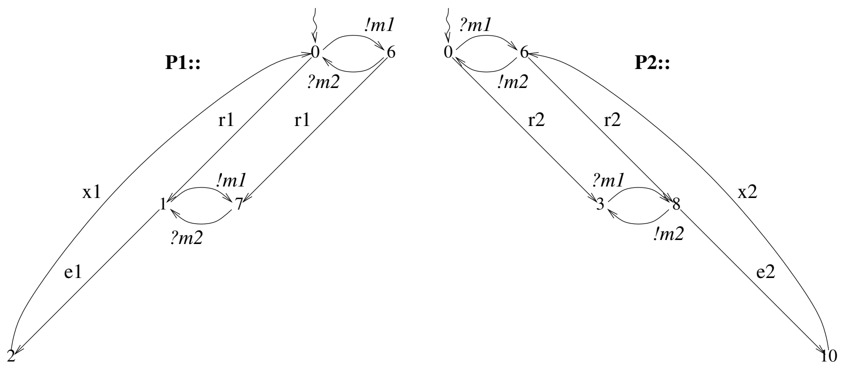


Fig. 12. Mutual exclusion: communicating automata generated from the second refinement.

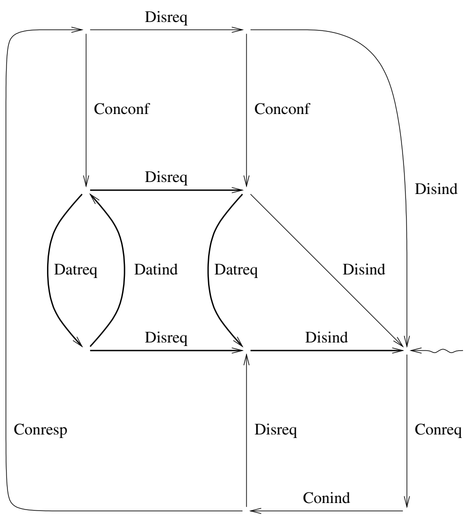


Fig. 13. The INRES protocol: specification of service.

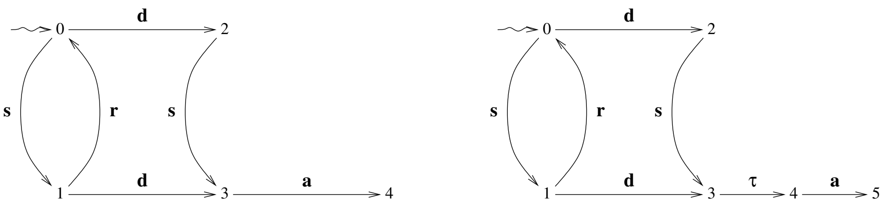


Fig. 14. Simple protocol: specification of service and its refinement.

Unfortunately the automaton specifying the service (shown on the left-hand side of Fig. 14) is not isomorphic to the marking graph of any distributable Petri net: event a and state 2 cannot be separated. However, this can be alleviated by refining the automaton into a weakly bisimilar automaton which is actually the marking graph of a distribuable Petri net.

In [CKL95, CKL98], is advocated an event-splitting heuristics for refining non-separated automata into separated automata. In our case, the two occurrences of s may be replaced by s_1 and s_2, and similarly for d, leading to three different refinements. However, none of the refined automata is separated with respect to the restricted set of *localisable* regions, compatible with the distribution constraints. Even though in the non-distributed case event splitting is a systematic method for refining automata into separated automata, it is potentially useless in the distributed case.

As an alternative resort, refinement can be done by inserting silent transitions while keeping weak

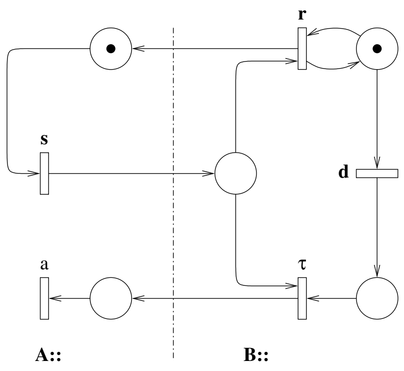


Fig. 15. Simple protocol: synthesised Petri net.

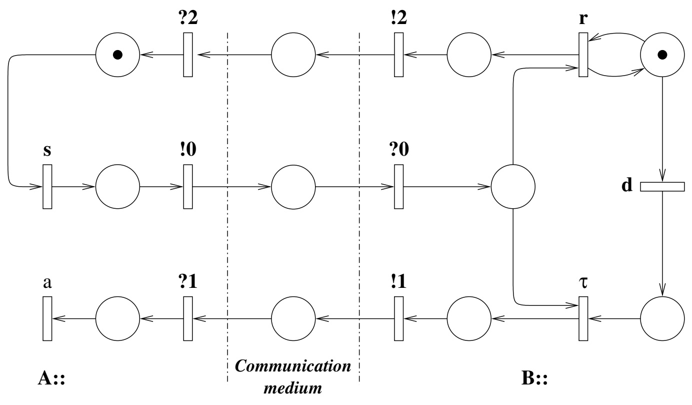


Fig. 16. Simple protocol: Petri net with communications.

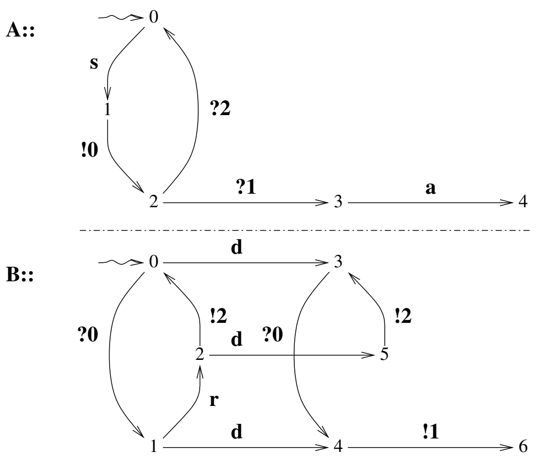


Fig. 17. Simple protocol: communicating automata.

bisimilarity. This has been used in [Yak98] for a similar purpose. In our case, the adequate refinement step consists in replacing transition $3 \xrightarrow{a} 4$ by two transitions: $3 \xrightarrow{\tau} 4 \xrightarrow{a} 5$ (see the automaton on the right-hand side of Fig. 14), with silent event τ located on process B. The refined automaton is then isomorphic to the reachable marking graph of the Petri net shown in Fig. 15. Places and transitions are sorted according to their locations: A on the left and B on the right. The distributable Petri net is then expanded into the Petri net shown in Fig. 16 by making communications between processes explicit. By distributing its reachable state graph as indicated in Section 5.2.2, with an abstraction from $\mathscr{E} = \{\tau\}$ and with minimisation, one obtains the automata shown in Fig. 17. It follows from the method that this distributed implementation is behaviourally correct. In spite of the minimisation, the automaton on the left-hand side of Fig. 14 can moreover be reconstructed from this distributed implementation. The full INRES protocol can be dealt with in much the same way; however, with an increased complexity.

7. Conclusion

A novel method for producing distributed implementations of finite reactive automata was presented in this paper, based on the synthesis of general Petri nets, that relies in turn on linear algebra. A tool called SYNET was built to this effect, integrating elementary algorithms on graphs and standard algorithms of linear algebra. Our limited experience with the application of this tool to the synthesis of distributed protocols makes us rather confident in the practical usefulness of the method. Nevertheless, different approaches to the distributed realisation of finite reactive automata may emerge, and we do not claim that the method based on distributable Petri nets will always give the best results. Notwithstanding this fact, two major problems remain still to be solved in the framework of distributable Petri net synthesis. The first problem, touched upon in Section 6, is to define techniques for transforming arbitrary automata into synthesisable automata. We saw that event splitting is not the right answer in the context of distribution. More promising is the investigation of silent event insertions, started by W. Vogler in the simpler context of one-safe nets and with a view at independence rather than distribution [Vog99]. The second problem, more general, is to re-examine Petri net synthesis for transition systems given by computational rules rather than by extension (i.e., as sets of states and transitions). One may take as examples non-expanded products of finite automata, or guarded expressions in some logical language, and more generally symbolic transition systems. Dealing with both types of problems would open large perspectives to the ideas exposed here.

References

[BBD95] Badouel, E., Bernardinello, L. and Darondeau, P.: Polynomial algorithms for the synthesis of bounded nets. *Proceedings Caap 95*, Lecture Notes in Computer Science, 915, pp. 647–679, 1995.

[BBD97] Badouel, E., Bernardinello, L., and Darondeau, P.: The synthesis problem for elementary net systems is NP-complete. *Theoretical Computer Science*, 186, pp. 107–134, 1997.

[BaD96] Badouel, E. and Darondeau, P.: On the synthesis of general Petri nets. *Inria Research Report* 3025, 1996.

[BDP96] Bernardinello, L., De Michelis, G., Petruni, K. and Vigna, S.: On the synchronic structure of transition systems. In J. Desel, editor, *Structures in Concurrency Theory*, pages 11–31. Workshops in Computing. Springer-Verlag, 1996.

[Cai97] Caillaud, B.: SYNET: un outil de synthèse de réseaux de Petri bornés, applications. *Irisa Research Report* 1101, 1997.

[Cai99] Caillaud, B.: Bounded Petri-net synthesis techniques and their applications to the distribution of reactive automata. *European Journal of Automation*, 33 (9,10), pp. 925–942, 1999.

[Che71] Chen, W. K.: *Applied Graph Theory*. North-Holland, 1971.

[CKL95] Cortadella, J., Kishinevsky, M., Lavagno, L. and Yakovlev, A.: Synthesising Petri nets from state-based models. In *Proceedings of ICCAD'95*, pages 164–173. IEEE Computer Society Press, 1995.

[CKL98] Cortadella, J., Kishinevsky, M., Lavagno, L. and Yakovlev, A.: Deriving Petri nets from finite transition systems. *IEEE Transactions on Computers*, 47 (8), pp. 859–882, 1998.

[CKK96] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L. and Yakovlev, A.: Complete state encoding based on the theory of regions. In *Proceedings of the 2nd International Symposium on Advanced Research on Asynchronous Circuits and Systems*, pages 36–47. IEEE Computer Society Press, 1996.

[DeR96] Desel, J. and Reisig, W.: The synthesis problem of Petri nets. *Acta Informatica*, 33, pp. 297–315, 1996.

[DrS93] Droste, M. and Shortt, R. M.: Petri Nets and automata with concurrency relations: an adjunction. In M. Droste and Y. Gurevich, editors, *Semantics of Programming Languages and Model Theory*, pages 69–87, 1993.

[DrS96] Droste, M. and Shortt, R. M.: From Petri nets to automata with concurrency. Draft communicated to the authors, 1996.

[EhR90a] Ehrenfeucht, A. and Rozenberg, G.: Partial (set) 2-structures. Part I: Basic notions and the representation problem. *Acta Informatica*, 27, pp. 315–342, 1990.

[EhR90b] Ehrenfeucht, A. and Rozenberg, G.: Partial (set) 2-structures. Part II: State spaces of concurrent systems. *Acta Informatica*, 27, pp. 343–368, 1990.

[vGW89] van Glabbeek, R. J. and Weijland, W. P.: Branching time and abstraction in bisimulation semantics. In *Proceedings of IFIP Congress*, pages 613–618. North-Holland/IFIP, 1989.

[GoM79] Gondran, M. and Minoux, M.: *Graphes et algorithmes*. Eyrolles, 1979. English translation: *Graphs and Algorithms*. John Wiley, 1984.

[Hog91] Hogrefe, D.: OSI-formal specification case study: the INRES protocol and service. *Technical Report* 91-012, University of Bern, 1991.

[Hop91] Hopkins, R.: Distributable nets. *Advances in Petri Nets 1991*, pages 161–187. Lecture Notes in Computer Science, 524, 1991.

[MaB67] Mac Lane, S. and Birkhoff, G.: *Algebra*. Chelsea Publishing Company, 1967.

[Muk92] Mukund, M.: Petri nets and step transition systems. *International Journal of Foundations of Computer Science*, 3 (4), pp. 443–478, 1992.

[Rei85] Reisig, W.: *Petri Nets*. EATCS Monographs on Theoretical Computer Science, volume 4. Springer-Verlag, 1985.

[RXG00] Rezg, N., Xie, X. and Ghaffari, A.: Supervisory control in discrete event systems using the theory of regions. In R. Boel and G. Stremersch, editors, *Discrete Event Systems: Analysis and Control*, pages 391–398 Kluwer Academic Publishers, 2000.

[Sch86] Schrijver, A.: *Theory of Linear and Integer Programming*. John Wiley, 1986.

[Synet] `http://www.irisa.fr/pampa/LOGICIELS/synet/synet.html`

[Tre96] Tretmans, J.: Test generation with inputs, outputs and repetitive quiescence. *Journal on Software: Concepts and Tools*, 17 (3), pp. 391–398, 1996.

[Vog99] Vogler, W.: Concurrent implementation of asynchronous transition systems. In *Proceedings of ICATPN'99*, pages 284–303. Lecture Notes in Computer Science, 1639, 1999.

[Yak98] Yakovlev, A.: Designing control logic for counterflow pipeline processor using Petri nets. *Formal Methods in System Design*, 12, pp. 39–71, 1998.

Received November 2000
Accepted in revised form August 2001 by D Simpson