Polynomial Algorithms for the Synthesis of Bounded Nets

Eric Badouel, Luca Bernardinello, Philippe Darondeau

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Polynomial Algorithms for the Synthesis of Bounded Nets

Eric Badouel , Luca Bernardinello and Philippe Darondeau

N˚ 2316

Juillet 1994

PROGRAMME 2

ISSN 0249-6399

Polynomial Algorithms for the Synthesis of Bounded Nets

Eric Badouel *, Luca Bernardinello* and Philippe Darondeau*

Programme 2 — Calcul symbolique, programmation et génie logiciel
Projet Micas

Rapport de recherche n° 2316 — Juillet 1994 — 27 pages

Abstract: The so-called synthesis problem for nets, which consists in deciding whether a given graph is isomorphic to the case graph of some net, and then constructing the net, has been solved in the litterature for various types of nets, ranging from elementary nets to Petri nets. The common principle for the synthesis is the idea of regions in graphs, representing possible extensions of places in nets. However, no practical algorithm has been defined so far for the synthesis. We give here explicit algorithms solving in polynomial time the synthesis problem for bounded nets from regular languages or from finite automata.

Key-words: Synthesis Problem for Nets, Petri Nets, Regions, Algorithms

(Résumé : tsvp)

This work was partly supported by the French P.R.C. *Modèles et Preuves*, by the H.C.M. Network *Express*, and by the H.C.M. fellowship granted to Luca Bernardinello, on leave from the University of Milan.

*Email:{Eric.Badouel, Luca.Bernardinello, Philippe.Darondeau}@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Algorithmes polynomiaux pour la synthèse de réseaux bornés

Résumé : Le problème de synthèse de réseaux, qui consiste à décider pour un graphe donné s'il est isomorphe à un graphe de marquage d'un réseau, est résolu pour diverses classes de réseaux allant des réseaux élémentaires aux réseaux de Petri. Le principe général est de caractériser les sous ensembles du graphe, appelés *régions*, qui représentent les extensions des places du réseau à synthétiser. Cependant la complexité théorique de la solution est exponentielle et aucune optimisation menant à des algorithmes utilisables n'a été présentée jusqu'à lors. Nous présentons ici des algorithmes explicites qui résolvent en temps polynomial le problème de synthèse de réseaux bornés à partir de langages réguliers ou à partir d'automates finis.

Mots-clé : Synthèse de réseaux, réseaux de Petri, régions, algorithmes

Contents

1 Introduction

Ehrenfeucht and Rozenberg addressed in [ER90] the problem of characterizing axiomatically the class of simple directed graphs labeled on arcs which may be represented unambiguously by a set of subsets of an overall coding set together with a set of mutual differences between them. Central to that work is the concept of *region* in a graph, defined as a set of vertices which are entered, exited, or left invariant by all arcs with an identical label. The overall coding set for a graph may always be chosen as its set of regions. A region is a property of the vertices it contains. Conversely, a vertex may be represented as the set of regions which it belongs to, and that representation is unambiguous if and only if every pair of vertices is *separated* in the usual sense by some region. Since all arcs with an identical label carry the same change of properties, a label may be represented as the mutual difference between the respective sets of properties

of the source and target vertices of any arc associated with that label in the graph, and that representation is unambiguous due to simpleness. The graph may be reconstructed from the set of mutual differences if each vertex which is not the origin of any arc with a given label is *separated* from that label by some region, which means that the region contains all the origins of the arcs carrying that label, but not the considered vertex. The so-called *regional axioms* specify those two different requirements of separation. Ehrenfeucht and Rozenberg applied the regional axioms to solve the synthesis problem for elementary nets in [ER90]. Given a simple directed graph labeled on arcs, they construct from that graph an elementary net with one place per region, one transition per label, and with flow relations between places and transitions laid down according to the mutual differences between vertices seen as sets of places, or markings. The case graph of the net assembled in this way is isomorphic to the given graph if and only if that graph is an *elementary transition system*, which implies essentially that the two regional axioms are satisfied. It is worth noting that in case the given graph is the case graph of an elementary net, the construction leads to a *saturated* version of that net, where extra places have been added without disturbing the behaviour but only copies of those places can still be added. Desel and Reisig observed in [DR92] that one may optimize the solution of the synthesis problem by computing any reduced set of regions sufficient to ensure satisfaction of the regional axioms in the graph. Along a similar line, Bernardinello proved in [Ber93] that the set of minimal regions (w.r.t. set inclusion) is adequate for that purpose. Unfortunately, this result does not seem to lead specially to practical algorithms with low complexity.

We will show that standard techniques of linear algebra lead to smooth algorithms for the synthesis of bounded nets in the extended framework of *general regions*, which were introduced under variant forms by Droste and Shortt [DS93], Mukund [Muk93], and Bernardinello, De Michelis and Petruni [BDP93]. General regions are multisets where regions are sets. A general region in a labeled graph is a multiset of vertices whose rank of membership is modified by a uniform translation along all arcs with an identical label. Since ranks of membership are whole numbers, uniform translations on ranks are defined by relative numbers. A general region is a property which may be satisfied at different degrees, measured by ranks of membership of vertices. In a dual way, a vertex may be represented as the multiset of properties which it

satisfies at some positive rank. Given a simple directed graph labeled on arcs, one can construct an induced place transition net with one place per general region, one transition per label, and with arcs between places and transitions weighted by norms of uniform translations and directed according to the signs of their defining numbers. Here again, the place transition net assembled in this way is a saturated net and its case graph is isomorphic to the given graph if and only if that graph satisfies the two regional axioms, restated in terms of ranked membership.

The object of this paper is to provide practical algorithms for deciding satisfaction of the regional axioms in a *finite* graph and for computing in that case just enough general regions to induce a place transition net with an isomorphic case graph. We leave for further study possible extensions to particular classes of infinite graphs. For sure, a finite graph is never the case graph of an unbounded Net. We shall therefore deal exclusively with bounded nets, whose precise definition is given in section 2.

We tackle in a first stage a relaxation of the synthesis problem for bounded nets, which is addressed in section 3 for languages defined by regular expressions. A language closed under left factors may be identified with a deterministic labeled tree. Regions in a language may therefore be defined as general regions in the underlying tree. Of special interest since we deal with bounded nets are general regions with bounded rank of membership, which we call *bounded regions*. The synthesis problem for languages is not quite the same as for labeled graphs: the sole constraint on the net assembled from regions in a language is to behave according to that language, which is significantly weaker than having a case graph isomorphic to the underlying tree. This weaker requirement is met if and only if that tree satisfies the second regional axiom, which was observed by Hoogers, Kleijn and Thiagarajan in the context of trace languages [HKT92]. A region in a language is totally determined by an offset, which is the rank of membership of the empty word, and by a vector of displacements, which are relative numbers defining the uniform translations attached to labels. Abstracting a little more, one may forget about offsets and consider *abstract regions* defined as vectors of relative numbers, or yet equivalently as morphisms from the free monoid over the set of labels to the additive monoid of relative numbers. The abstract regions which are the projections of the bounded regions in a regular language form a module of finite dimension

over $\mathbb{Z}$. We will show that deciding whether a regular language satisfies the second regional axiom and computing in that case a bounded net with equivalent behaviour may be reduced to solving finite linear systems over the integers, uniform in the regular expression. We evolve therefrom a practical method for the synthesis of bounded nets, based on *polynomial* algorithms discovered in the late seventies for linear programming over the rational field. Altogether, we obtain a polynomial algorithm for the synthesis of nets from regular expressions in the so-called failures tree form (which in particular means they are represented by trees labeled with action symbols and starred expressions).

We tackle in a second stage the general synthesis problem for bounded nets, which is addressed in section 4 for finite automata. Since we are back in the framework of labeled graphs, the first regional axiom is re-imposed. A general region in a finite automaton is determined by an offset, which is the rank of membership of the initial state, and by a vector of relative numbers which associate uniform translations to labels. A region in a finite automaton is always a region in the language of the automaton, but the converse is not true. The set of abstract regions in a finite automaton is a submodule of the module of abstract regions in the language of the automaton. The abstract regions which are projections of bounded regions are the solutions of a system of linear equations which express equality constraints relative to states, stronger in general than the boundedness constraints relative to the language of the automaton. Altogether, one retrieves the constraints set on *synchronic distances* in [BDP93], which shows that abstract regions are in a one-one correspondence with synchronic distances. The advantage of the more algebraic view taken here is to lead directly to algorithms. We will show that deciding whether a finite automaton satisfies the two regional axiom and then computing a bounded net with equivalent behaviour, reduce again to solving finite linear systems over the rational field, uniform in the automaton. Altogether, we obtain a polynomial algorithm for the synthesis of nets from finite automata.

2 Notations for Petri Nets

We adopt in this paper a notation for Place/Transition nets slightly different from the usual one; this should not cause confusion.

Definition 2.1 (Nets) *A* net *is a structure $N = (P, T, W)$, where P and T are disjoint sets, of* places *and* transitions *respectively, and $W : P \times T \to \mathbb{Z}$ is the* weight *function. The set of* input *(resp.* output*) places of a transition t is the set ${}^\bullet t$ (resp. $t^\bullet$) of places p such that $W(p,t) < 0$ (resp. $W(p,t) > 0$). A* marking *of N is a map $m : P \to \mathbb{N}$.*

> Nets are usually equipped with a flow relation $F \subseteq (P \times T) \cup (T \times P)$ and a weight function $W : F \to \mathbb{N}^+$. We can adopt here a more compact notation because we consider exclusively *pure* nets, i.e. nets such that $\forall x, y \in P \cup T \quad (x,y) \in F \Rightarrow (y,x) \notin F$.

The overall behaviour of a Place/Transition net is determined by the so-called *firing rule* which tells that a transition is enabled at a marking if that marking supplies enough resources in each of its input places.

Definition 2.2 (Firing rule) *Let $N = (P, T, W)$ be a net and M a marking of N; a transition $t \in T$ is* enabled *at m if $\forall p \in P \quad M(p) + W(p,t) \geq 0$. If transition t is enabled at marking M, then it can fire; in doing so, it produces a new marking M', defined by $\forall p \in P \quad M'(p) = M(p) + W(p,t)$. This firing step is denoted $M[t>M'$. A sequence of transitions $u = t_1 t_2 \cdots t_n$ is enabled at a marking M_0 if $\exists M_1, \ldots M_n$ such that $M_{i-1}[t_i > M_i$, for all $i = 1, \ldots, n$. This firing sequence is denoted $M_0[u>M_n$.*

While the overall behaviour of a Place/Transition net reflects its whole structure, one is most often interested in restricted behaviours induced by initial markings.

Definition 2.3 (Marked net) *A* marked *net is a structure $N = (P, T, W, M_0)$, where (P, T, W) is a net and $M_0 : P \to \mathbb{N}$ is the* initial marking*. The set of* reachable markings *of N is the smallest set $\mathcal{M}(N)$ of markings such that*

1. *$M_0 \in \mathcal{M}(N)$,*
2. *if $M \in \mathcal{M}(N)$ and $M[t>M'$ then $M' \in \mathcal{M}(N)$.*

The marking graph *of N is the labeled transition system $ng(N)$ with set of states $\mathcal{M}(N)$, initial state M_0, set of labels T, and set of labeled transitions $Tr(N) \subseteq \mathcal{M}(N) \times T \times \mathcal{M}(N)$ defined by*
$(M, t, M') \in Tr(N) \Leftrightarrow M[t>M'$. The language *of net N, or equivalently the language of the marking graph $ng(N)$, is the set of all sequences of transitions enabled at M_0 (thus this language is prefix closed).*

Of special interest in this paper are marked nets with finite sets of reachable markings. A relaxed form of finiteness may be defined as follows.

Definition 2.4 (Bounded net) *A place p of a marked net $N = (P, T, W, M_0)$ is* bounded *if the set $\{M(p) | M \in \mathcal{M}(N)\}$ is finite. A marked net with bounded places is called a* bounded *net.*

Droste and Shortt observed in [DS92] that a bounded net with finitely many transitions must have a finite set of reachable markings. Hence any bounded net with finite dimension may be transformed into a finite equivalent net.

3 Regions in Regular Languages

The goal of the section is to solve the following problem:

Problem 3.1 *Given a prefix-closed language $L \subseteq A^*$, decide whether there exists a marked net with language L and if so, construct such a net.*

We will present a polynomial algorithm which answers this problem for regular languages. The basic constituent of the proposed solution is the principle of abstract regions which may in fact be applied to arbitrary languages. We shall therefore cast the definition of regions in a general context, and subsequently focus on regular languages.

Throughout the section, let L denote a *prefix-closed* language over a finite alphabet $A = \{a_1, \ldots, a_n\}$, ranged over by a, and for any map $\rho : A \to \mathbb{Z}$, let $\hat{\rho} : A^* \to \mathbb{Z}$ denote the monoid morphism $\hat{\rho}(ua) = \hat{\rho}(u) + \rho(a)$. Thus in particular $0 \in \hat{\rho}(L)$, where $\hat{\rho}(L) = \{\hat{\rho}(u) \mid u \in L\}$.

3.1 Regions in Formal Languages

An abstract region in a prefix-closed language L over alphabet A is a morphism from A^* to $\mathbb{Z}$, satisfying the requirement that L should be mapped to an interval of $\mathbb{Z}$ bounded from below. If one imagines the morphic image of a word in L as the measure of variation of an abstract resource through a deterministic process, the requirement guarantees that starvation may be avoided

globally by feeding in every process in L with a fixed amount of initial resource. Bounded abstract regions, which will play an essential role in our study of regular languages, satisfy the stronger requirement that L should be mapped to a finite interval of $\mathbb{Z}$, reflecting the intuition that a language considered as a non-deterministic process can only produce a bounded amount of resource.

Definition 3.2 (Regions of a language) *A* region *of L is a map $\rho : A \to \mathbb{Z}$ such that the set $\{\hat{\rho}(w) \mid w \in L\}$ has an infimum. A* bounded *region of L is a region ρ of L such that the set $\{\hat{\rho}(w) \mid w \in L\}$ has also a supremum. Let R_L, resp. BR_L denote the set of regions, resp. bounded regions of L.*

The algebraic properties of abstract regions which follow immediately from the definition are gathered below.

Fact 3.3 *The sets R_L and BR_L are closed under pointwise sum of maps, and the null region ρ_0 with constant value $\rho_0(a) = 0$ is the neutral element for sum. Moreover, the set BR_L is closed under inverses. Thus R_L is a monoid, and BR_L is an abelian group.*

As already observed in [BDP93], the bounded regions of a language over alphabet A are a subgroup of the free abelian group of maps from A to $\mathbb{Z}$, which is finitely generated since A is finite. As a consequence, BR_L is also a free group with a finite set of generators, which is tantamount to a module of finite dimension over $\mathbb{Z}$. In order to recover *concrete* regions from *abstract* regions, it suffices to attach to each abstract region $\rho \in R_L$ the minimal *offset* $M_L(\rho)$ such that any possible displacement $\hat{\rho}(u)$ associated with some word $u \in L$ leads therefrom to a non negative value $M_L(\rho) + \hat{\rho}(u)$. The term *region* is used in all the sequel as an abbreviation for *abstract region*.

We are now in a position to derive marked nets from prefix-closed languages: the transitions are the symbols in the alphabet, the places are regions, and the weight functions are set in agreement with the values of regions.

Definition 3.4 (Nets derived from languages)

1. *The* saturated net $\mathcal{N}(L)$ *derived from L is the net (R_L, A, W, M_L), where $W(\rho, a) = \rho(a)$ for $\rho \in R_L$, and $M_L(\rho) = -\inf\{\hat{\rho}(u) \mid u \in L\}$.*

2. *For any set of regions $R \subseteq R_L$, the* R-net *derived from L is the sub-net of $\mathcal{N}(L)$ with set of places R.*

3. *In particular, the* saturated bounded net *$\mathcal{BN}(L)$ derived from L is the sub-net of $\mathcal{N}(L)$ with set of places BR_L.*

We are mostly interested in determining the necessary and sufficient conditions under which the languages of the nets $\mathcal{N}(L)$ or $\mathcal{BN}(L)$ coincide with the language L. Those conditions may be stated in terms of the following properties of separation.

Definition 3.5 (Separation properties for languages) *Let $R \subseteq R_L$ be a subset of regions of L, then L is said to be* separated by R *if and only if for all word u, $(u \notin L \Rightarrow \exists \rho \in R \quad M_L(\rho) + \hat{\rho}(u) < 0)$. The language L is* separated *if it is separated by R_L, and* boundedly separated *if it is separated by BR_L.*

Remark 3.6 *We say that a region ρ of L kills the word ua whenever $M_L(\rho) + \hat{\rho}(ua) < 0$ whereas $M_L(\rho) + \hat{\rho}(u) \geq 0$. Thefore a prefix-closed language L is separated by R if and only if every* faulty word *$u \cdot a \in (L \cdot A) \setminus L$ is killed by some region of R.*

Example 3.7

1. *Let L be the (prefix-closed) regular language $a^* + a^*b$ then necessarily $\rho(a) \geq 0$ for every region ρ of L, thus no region $\rho \in R_L$ kills the word ba and L is not separated (since $b \in L$ and $ba \notin L$).*

2. *Let L be the (prefix-closed) regular language $a^* + a(a^*)b(a^*)$ then L is separated, but L is not boundedly separated, since any region ρ of L which kills the word $b \notin L$ satisfies $\rho(a) > 0$ (any region ρ of L must satisfy $M_L(\rho) + \hat{\rho}(ab) \geq 0$).*

3. *Let L be the prefix-closure of the language $\{(a.\overline{a})^n b(c.\overline{c})^n | n \in I\!N\}$, which is algebraic but not regular, then L is separated by a finite set of regions but it cannot be separated by any set of bounded regions (if it was, L would be regular according to the remark of Droste and Schortt).*

The places of a net may always be identified with regions in the language of that net, moreover the language of a net is separated by the set of regions associated with places in the net, even though $M_L(\rho)$ does not necessarily coincide with the value of the initial marking of the net at the place associated with ρ.

Proposition 3.8 *A prefix-closed language L coincides with the language of the R-net derived from L iff it is separated by R.*

Proof: Let notations as follows: $N(L,R)$ denotes the R-net derived from language L, M_R denotes the initial marking of $N(L,R)$, that is M_L restricted to R, and $\mathcal{L}(N)$ denotes the language of net N. Observe that, by definition, the language of a net is prefix-closed. In order to establish the proposition, we will use a lemma which follows straightforwardly from the definition of $N(L,R)$:

Lemma 3.9 *Let $u \in \mathcal{L}(N(L,R))$, then for each $\rho \in R$, $M_R[u>M$ implies $M(\rho) = M_R(\rho) + \hat{\rho}(u)$.*

With the help of that lemma, we prove that $L \subseteq \mathcal{L}(N(L,R))$ for any subset R of regions of L. The proof is by induction on the length of $u \in L$. If $|u| = 0$, i.e. $u = \epsilon$, then clearly $u \in \mathcal{L}(N(L,R))$. Suppose that $u \in L$ implies $u \in \mathcal{L}(N(L,R))$ for $|u| \leq n$, and let $u.a \in L$, with $|u| = n$ and $a \in A$. By definition of regions in L, $M_R(\rho) + \hat{\rho}(u) + \rho a \geq 0$ for any $\rho \in R_L$, and by induction hypothesis, $u \in \mathcal{L}(N(L,R))$, hence $M_R[u>M$ where $M(\rho) = M_R(\rho) + \hat{\rho}(u)$ and the above inequality shows that a is enabled at M. We prove now that $L = \mathcal{L}(N(L,R))$ if L is separated by R. Let $u \in L$, then by the first part of the proof, $u \in \mathcal{L}(N(L,R))$, and by definition of separatedness:

$$u.a \in L \Leftrightarrow [\forall \rho \in R \;\; M_R(\rho) + \hat{\rho}(u.a) \geq 0] \Leftrightarrow \;\; a \text{ is enabled at } M,$$

where M is the marking such that $M_R[u>M$ in $N(L,R)$, and therefore $L = \mathcal{L}(N(L,R))$. We prove finally that $L \neq \mathcal{L}(N(L,R))$ if L is not separated by R. If so, by definition of separatedness, there must exist $u \in L$ and $a \in A$ such that $u.a \notin L$ although $M_R(\rho) + \hat{\rho}(u.a) \geq 0$ for every $\rho \in R$. But in that case, $u.a \in \mathcal{L}(N(L,R))$, and therefore $L \neq \mathcal{L}(N(L,R))$. ∎

Corollary 3.10

1. *A prefix-closed language L coincides with the language of the saturated net $\mathcal{N}(L)$ iff it is separated.*

2. *A prefix-closed language L coincides with the language of the saturated bounded net $\mathcal{BN}(L)$ iff it is boundedly separated.*

Observe that no place (except places obtained from existing places by possibly increasing their values at the initial marking) can be added to the saturated net $\mathcal{N}(L)$ derived from a separated language L without modifying its behaviour (and similarly for bounded places w.r.t saturated bounded nets and boundedly separated languages). In the rest of the section, we concentrate on the problem of synthesizing bounded nets from regular expressions.

3.2 Computing a Basis for Bounded Regions in a Regular Language

We already observed (see Fact. 3.3) that the bounded regions of a prefix-closed language form a module of finite dimension over $\mathbb{Z}$. Our purpose is now to design an algorithm which computes in polynomial time a basis of that module for any prefix-closed and regular language given by a regular expression. Let us fix the notation.

Definition 3.11 (Regular expressions) *A* regular expression *over alphabet A is an expression E in the B.N.F. syntax $E ::= \varepsilon \mid a \mid E + E \mid E \times E \mid E^*$, where $a \in A$.*

Regular expressions E are mapped to regular languages $|E|$ by the obvious morphism $|\cdot|$, which is onto. By abuse of notations, we shall often extend to regular expressions notations defined for their languages, for instance BR_E will stand for the module of bounded regions of the language $|E|$. From now on, E is a fixed regular expression denoting a prefix-closed language. We will show that the module of regions BR_E is the kernel of a linear system over $\mathbb{Z}$, obtained by decomposing E into so-called *cyclic factors*, next translated independently to equations whose unknown are the values $\rho(a_1)...\rho(a_n)$ of an abstract region.

Definition 3.12 (Cyclic factors of a regular expression) *A* cyclic factor *of the regular expression E is the image of an* iterated *sub-expression F of E (i.e. such that F^* appears in F) by the* shortcut operator $\cdot^{@}$ *with inductive definition as follows: $(E^*)^{@} = \varepsilon$, $(E \times E')^{@} = (E)^{@} \times (E')^{@}$, $(E + E')^{@} = (E)^{@} + (E')^{@}$, $a^{@} = a$, $\varepsilon^{@} = \varepsilon$. Let $CF(E)$ denote the set of cyclic factors of E.*

Observation 3.13 *The sum of sizes of the cyclic factors of E is bounded from above by the size of E.*

Proposition 3.14 *A map $\rho : A \rightarrow \mathbb{Z}$ is a bounded region of the language defined by E iff $\hat{\rho}(E_i) = \{0\}$ for every cyclic factor E_i of E.*

> *Proof:* Let $\rho \in BR_E$, and F be an iterated subexpression of E. Let $u \in |F^{@}|$. Then $|E|$ contains for each $n \in \mathbb{N}$, a word of the form $u_1.u^n.u_2$, where $u_1, u_2 \in A^*$. Then $\hat{\rho}(u) = 0$ since ρ is bounded. Conversely, we first notice that if $\hat{\rho}(E_i) = \{0\}$ for each cyclic factor of E, then $\hat{\rho}(F) = \{0\}$ for each iterated subexpression of E. Hence, to evaluate $\hat{\rho}(u)$ for $u \in |E|$, we can delete segments of u generated by iterated subexpressions of E. What is left is an element of $E^{@}$; since the language $|E^{@}|$ is finite, $\hat{\rho}(E)$ is a bounded set. ∎

Proposition 3.14 shows that the module BR_E of bounded regions of E is the kernel $Ker(M_E)$ of a linear map $M_E : \mathbb{Z}^n \rightarrow \mathbb{Z}^n$, operating on n-vectors $\vec{\rho} = < \rho(a_1), ..., \rho(a_n) >$ which represent maps $\rho : A \rightarrow \mathbb{Z}$, where $A = \{a_1 ... a_n\}$. For each cyclic factor E_i of E, the condition $\hat{\rho}(E_i) = \{0\}$ laid down by the proposition may in fact be translated in polynomial time to an equivalent condition $M_i(\vec{\rho}) = \vec{0}$, where M_i is a linear map from $\mathbb{Z}^n$ to $\mathbb{Z}^{m_i}$ for some finite dimension m_i bounded from above by the size of E_i. The overall dimension $m = \sum\{m_i | E_i \in CF(E)\}$ is then bounded from above by the size of E.

> We give here the sketch of a polynomial algorithm computing the integer matrix M_i from the cyclic factor E_i. Let $E_{i,j}$ be an enumeration of the regular sub-expressions of E_i, where $E_i = E_{i,0}$ and $0 \leq j \leq p_i$, and let $\{x_j \mid 0 \leq j \leq p_i\}$ be a set of integer variables. Each variable x_j is aimed at representing the contents of the corresponding set $\hat{\rho}(E_{i,j})$,

which must be a singleton set in order that condition $\hat{\rho}(E_i) = \{0\}$ may be satisfied. Let $\{y_k \mid 1 \leq k \leq n\}$ be another set of integer variables, aimed at representing the values $\rho(a_k)$. By iterating the following step indexed by j, one assembles a system of linear equations in variables x_j and y_k.
For each $j \in \{0, \ldots, p_i\}$:

1. if $E_{i,j} = \varepsilon$, write the equation $x_j = 0$;
2. if $E_{i,j} = a_k$, write the equation $x_j = y_k$;
3. if $E_{i,j} = E_{i,j_1} \times E_{i,j_2}$, write the equation $x_j = x_{j_1} + x_{j_2}$;
4. if $E_{i,j} = E_{i,j_1} + E_{i,j_2}$, write the pair of equations $x_j = x_{j_1}$ and $x_j = x_{j_2}$

Let the equation $x_0 = 0$ be added to the system assembled in this way. The operator M_i is finally obtained by gaussian elimination of the variables x_j.

Once an $(m_i \times n)$ matrix M_i such that $M_i\ \vec{\rho} = \vec{0}$ iff $\hat{\rho}(E_i) = \{0\}$ has been obtained for each cyclic factor E_i of E, it only remains to pile up all the matrices M_i in order to form an $(m \times n)$ matrix M_E satisfying $BR_E = \{\vec{\rho} \mid M_E\vec{\rho} = \vec{0}\}$. At this stage, the algorithm of von zur Gathen and Sieveking (see [Sch86] p.58) may be used to compute in polynomial time a family of vectors $\vec{\rho}_1 \ldots \vec{\rho}_t$ of integers such that

$$\{\vec{\rho} \mid M_E\vec{\rho} = \vec{0}\} \ = \ \{\lambda_1\ \vec{\rho}_1 + \ldots + \lambda_t\ \vec{\rho}_t \mid \lambda_1, \ldots, \lambda_t \in \mathbb{Z}\}$$

with $\vec{\rho}_1 \ldots \vec{\rho}_t$ linearly independent. In the sequel, $\{\vec{\rho}_1 \ldots \vec{\rho}_t\}$ is assumed to be a fixed basis of the module of bounded regions of the language defined by the expression E.

3.3 Deciding upon Separatedness of a Regular Language

Resting upon the availability of computable bases for modules of regions, we now intend to design an algorithm deciding whether a prefix-closed and regular language is boundedly separated or equivalently, whether it coincides with the

language of a bounded net. We do not know how solving this problem in polynomial time for arbitrary regular expressions. We shall therefore focus on a special form of regular expressions, allowing to decide on separatedness in polynomial time. These expressions may be seen as trees, labeled on arcs with action symbols or starred expressions, such that any restriction of a tree T to a particular branch induces a regular language where all words are failure equivalent in $|T|$.

Definition 3.15 (Regular expressions in tree form) *A regular expression in* tree form *is a regular expression T written according to the restricted syntax*

$$T ::= \varepsilon \mid a \times T \mid E^* \times T \mid T + T,$$

where E is a regular expression. The branches *of T are the regular expressions in the set $br(T)$ defined as*

$$br(\varepsilon) = \{\varepsilon\} \quad br(T_1 + T_2) = br(T_1) \cup br(T_2)$$
$$br(a \times T) = \{a \times B | B \in br(T)\} \quad br(E^* \times T) = \{E^* \times B \mid B \in br(T)\}$$

Clearly, every regular expression may be set in tree form, but that transformation may induce an exponential increase in size.

> Consider for instance the indexed family of regular expressions $E_n = (\varepsilon + a + b) \times \ \ldots \ \times (\varepsilon + a + b)$, where E_n contains n occurrences of the symbol a (or b). The language $|E_n|$ contains exactly the words of length less than or equal to n over the alphabet $\{a, b\}$. The tree-like expansion of E_n contains $2^n - 1$ occurrences of the symbol a (or b), and the minimal words on two letters which do not belong to $|E_n|$, i.e. the words of length $n + 1$, are 2^{n+1} in number. Our algorithm will take one step for each of these faulty words, and thus altogether 2^{n+1} steps, for deciding on separatedness, when the constant region $\rho(a) = \rho(b) = -1$ with offset $M_L(\rho) = n$ suffices to separate $|E_n|$!

Definition 3.16 (Regular expressions in failures tree form) *A regular expression in* failures tree *form is a regular expression T in tree form such that*

$$\forall B \in br(T) \ \ \forall u, u' \in |B| \ \ \forall a \in A \ \ u \cdot a \in |T| \Leftrightarrow u' \cdot a \in |T|$$

.

One can naturally decide whether a regular expression is in failures tree form, but the decision algorithm is exponential. In order to show that every regular expression may be set in failures tree form, we sketch below an (exponential) algorithm which produces such expressions from deterministic finite automata.

Let $\mathcal{A} = (Q, A, T, q_0, F)$ be a deterministic automaton, with initial state $q_0 \in Q$ and final states $F \subseteq Q$, recognizing a regular language L. For $q, q' \in Q$ and $R \subseteq Q$, let $L^R_{q,q'}$ denote the set of non-empty words $a_1 \ldots a_n$ labeling sequences of transitions $q_0 \stackrel{a_1}{\to} q_1 \stackrel{a_2}{\to} q_2 \ldots q_{n-1} \stackrel{a_n}{\to} q_n$ such that $q_0 = q$, $q_i \in R$ for $1 \leq i \leq n-1$, and $q_n = q'$. Similarly, let L^R_q denote the set which contains the empty word if $q \in F$, and in any case the non-empty words $a_1 \ldots a_n$ labeling sequences of transitions $q_0 \stackrel{a_1}{\to} q_1 \stackrel{a_2}{\to} q_2 \ldots q_{n-1} \stackrel{a_n}{\to} q_n$ such that $q_0 = q$, $q_i \in R$ for $1 \leq i \leq n-1$, and $q_n \in F \cap R$. Then the following equations

$$
\begin{aligned}
L &= L^Q_{q_0} \\
L^R_q &= (L^{R\setminus\{q\}}_{q,q})^* \cdot L^{R\setminus\{q\}}_q \qquad \text{if } q \in R \\
L^R_q &= \begin{cases} \varepsilon + \displaystyle\sum_{q \stackrel{a}{\to} q',\ q' \in R} a' \cdot L^R_{q'} & \text{if } q \in F \\ \displaystyle\sum_{q \stackrel{a}{\to} q',\ q' \in R} a' \cdot L^R_{q'} & \text{otherwise} \end{cases} \qquad \text{if } q \notin R
\end{aligned}
$$

produce a regular expression in failures tree form for language L, with branches

$$(L^{R_0}_{q_0,q_0})^* \cdot a_1 \cdot (L^{R_1}_{q_1,q_1})^* \cdot a_2 \ldots (L^{R_n}_{q_n,q_n})^* \varepsilon$$

such that $q_0 \stackrel{a_1}{\to} q_1 \stackrel{a_2}{\to} q_2 \ldots q_{n-1} \stackrel{a_n}{\to} q_n$, $q_{i+1} \in R_i = Q \setminus \{q_0, \ldots, q_{i-1}\}$, and $q_n \in F$.

The main interest of the failures tree form lies in the next two propositions. From now on, let T be a fixed regular expression in failures tree form, denoting a prefix-closed language over $A = \{a_1, \ldots, a_n\}$.

Proposition 3.17 *$\{\hat{\rho}(w) \mid w \in T\} = \{\hat{\rho}(w) \mid w \in T^{@}\}$ for every bounded region ρ of T (where $\cdot^{@}$ is the shortcut operator, see Def. 3.12).*

Proof: From Prop. 3.14, $\hat{\rho}(E_i) = \{0\}$ for each cyclic factor E_i of E, therefore, for each $u \in |T|$, we can find $u' \in |T^{@}|$ such that $\hat{\rho}(u) = \hat{\rho}(u')$. On the other hand $T^{@} \subseteq T$, where the regular expressions are viewed as languages. ∎

Proposition 3.18 *T is boundedly separated iff every* faulty word $w \in (T^{@}A \setminus T)$ *is killed by some bounded region of T.*

Proof: Assume that T is boundedly separated and let $w \in T^{@}A \setminus T$; then $w = u.a$, where $u \in T^{@}(\subseteq T)$ and $a \in A$. Since T is boundedly separated, and since $u \in T$ and $u.a \notin T$, there must exist some region $\rho \in BR_T$ that kills $u.a$. Conversely, assume that T is not boundedly separated, and let $u \in T$ and $a \in A$ be such that $u.a \notin T$ but no bounded region in T kills $u.a$. Let $B \in br(T)$ be a branch of T such that $u \in |B|$, and let u' be a word obtained by erasing the occurrences of the outermost starred subexpressions of B in u (there may be several possible words u', as u may belong to $|B|$ in more than one way). Then $u' \in T^{@}$ and $\hat{\rho}(u) = \hat{\rho}(u')$ for every bounded region ρ of T, hence no bounded region kills $u'.a$. However, $u'.a \notin T$, since otherwise $u.a$ would belong to T by definition of regular expressions in failures tree form. Hence $u'.a$ is a faulty word, but it is not killed by any bounded region. ∎

Let $\{u_h \mid 1 \leq h \leq H_T\}$ be an enumeration of $T^{@}$, and let $\{v_k \mid 1 \leq k \leq K_T\}$ be an enumeration of $(T^{@}A \setminus T)$. H_T and K_T are clearly bounded from above by linear functions of the size of T. In view of Def. 3.5, a bounded region $\rho \in BR_T$ kills a faulty word v_k iff $M_T(\rho) + \hat{\rho}(v_k) < 0$, with $M_T(\rho) = -\inf\{\hat{\rho}(w) \mid w \in T\}$. In view of Prop. 3.17, $M_T(\rho) = -\inf\{\hat{\rho}(u_h) \mid u_h \in T^{@}\}$. Thus, by Prop. 3.18, T is boundedly separated iff the following problem may be solved for every k (with $1 \leq k \leq K_T$):

Problem 3.19 *Find a linear combination $\vec{\rho} = \lambda_1 \vec{\rho}_1 + \ldots + \lambda_t \vec{\rho}_t$ of the base vectors of the module BR_T such that $\hat{\rho}(v_k) - \hat{\rho}(u_h) < 0$ for every h (with $1 \leq h \leq H_T$).*

We address now the above problem for a fixed word $v_k \in (T^{@}A \setminus T)$. Let $\vec{x} =< x_1, \ldots x_n >$ be the Parikh image of v_k, where x_i counts the occurrences of a_i. Similarly, let $\vec{y}_h$ be the Parikh image of u_h. Each condition $\hat{\rho}(v_k) - \hat{\rho}(u_h) < 0$ induces a corresponding constraint on the unknown $\vec{\Lambda} =< \lambda_1, \ldots, \lambda_t > (\in \mathbb{Z}^t)$, viz. the inequation: $\lambda_1 \ \vec{\rho}_1 \ (\vec{x} - \vec{y}_h) + \ldots + \lambda_t \ \vec{\rho}_t \ (\vec{x} - \vec{y}_h) < \vec{0}$. Assembling the constraints for $1 \leq h \leq H_T$, one obtains a linear system

$$M_k \vec{\Lambda} \leq (-1)^{H_T} \tag{1}$$

where $\vec{M}_k$ is an integral matrix and $(-1)^{H_T} =< -1, \ldots, -1 > (\in \mathbb{Z}^{H_T})$. We claim that (1) has an integral solution iff it has a rational solution. A rational solution $\vec{\Lambda} =< z_1/n_1, \ldots, z_t/n_t >$ gives indeed rise to an integral solution $n\vec{\Lambda}$ for any common multiple n of the denominators n_i. At this stage, the method of Khachiyan (see [Sch86] p.170) may be used to decide the feasability of (1) and to compute an explicit solution, if it exists, in polynomial time. Thus, every instance of Prob. 3.19 is solved explicitly, or shown unfeasible, in polynomial time.

Now, deciding whether T is boundedly separated takes polynomial time, since it reduces to solving K_T instances of Prob. 3.19.

3.4 Computing a Finite Net from a Regular Language

Let T be a boundedly separated, prefix-closed, and regular language. Thus, every faulty word $v_k \in (T^{@}A \setminus T)$ is killed by some region $\rho'_k \in BR_T$. Let $R = \{\rho'_k \mid 1 \leq k \leq K_T\}$, where ρ'_k kills v_k. In view of Prop. 3.8, T coincides with the language of the R-net derived from L, i.e. the finite net (R, A, W, M) with $W(\rho'_k, a) = \rho'_k(a)$ and $M(\rho'_k) = -\inf\{\hat{\rho}'_k(u_h) \mid 1 \leq h \leq H_T\}$. Hence the results of the section may be summarized as follows.

Theorem 3.20 *Let T be a prefix-closed language given by a regular expression in failures tree form, then one may decide whether $|T|$ coincides with the language of some finite net and construct that net in polynomial time.*

4 Regions in Finite Graphs

The goal of the section is to solve the following problem:

Problem 4.1 *Given an A-labeled graph G, decide whether there exists a marked net with marking graph isomorphic to G and if so, construct such a net.*

We will present a polynomial algorithm answering this problem for finite graphs. The basic constituent of the proposed solution is again the principle of abstract regions, but regions in graphs are more tightly constrained than regions in languages, as will appear soon. Since equality of languages is strictly weaker than isomorphism of graphs, the synthesis problem for languages is a weakening of the synthesis problem for graphs. Nevertheless, we do not know any polynomial reduction of one problem to the other. We propose therefore two variant but not directly related algorithms for solving the two different synthesis problems.

The principle of regions applies more generally to infinite graphs. We shall therefore introduce regions for general graphs, and then focus on finite graphs.

4.1 Some Graph Terminology

Definition 4.2 (Graphs) *A* graph $G = [S, T]$ *is given by a set* S *of* nodes, *a set* T *of transitions (or arcs), together with two maps* $\partial^0, \partial^1 : T \to S$, *indicating respectively the* source *and* target *of transitions. A* path *in* G *is a finite sequence of transitions* $t_1 \ldots t_k$ *where* $\partial^1(t_i) = \partial^0(t_{i+1})$ *for* $i < k$. *Nodes* $\partial^0(t_1)$ *and* $\partial^1(t_k)$ *are the* extremities, *or respectively the* initial *and* terminal *nodes of the path. A* chain *in* G *is a path in the graph* $[S, T + T^{-1}]$, *where* $\partial^0(t^{-1}) = \partial^1(t)$ *and* $\partial^1(t^{-1}) = \partial^0(t)$ *for* $t \in T$. *A* cycle *is a chain with identical extremities. A cycle which is a path is* directed. *A* rooted *graph is a graph with a distinguished node, called the* initial *node. Paths from the initial node are* initial *paths. An* accessible *graph is a rooted graph in which every node, to the possible exception of the initial node, is the terminal node of some initial path.*

Definition 4.3 (Labeled Graphs) *A* labeled graph $G = [S, T, A, l]$ *is a graph* $[S, T]$ *enriched with a* labeling *function* $l : T \to A$, *where* A *is the set of* actions. *A labeled graph is* deterministic *if* $(\partial^0(t) = \partial^0(t') \wedge l(t) = l(t')) \Rightarrow \partial^1(t) = \partial^1(t')$. *The* label *of chain* c, *denoted* $l(c)$, *is the sequence of actions* a *and* inverse *actions* a^{-1} *labeling arcs on that chain. The* Parikh image *of a chain* c, *denoted* $\pi(c)$, *is the evaluation of its label* $l(c)$ *in the commutative group* $F_{CG}(A)$ *freely generated by* A. *The* language $L(G) \subseteq A^*$ *of graph* G *is the set of labels of initial paths in* G.

Observe that $F_{CG}(A)$ can be identified with the set of maps $\rho : A \to \mathbb{Z}$ equipped with pointwise sum, which permits to represent the elements of the commutative group as A-indexed vectors of integers. Henceforth, for any map $\rho : A \to \mathbb{Z}$, let $\tilde{\rho} : F_{CG}(A) \to \mathbb{Z}$ denote the group morphism $\tilde{\rho}(ua) = \tilde{\rho}(u) + \rho(a)$, where $u \in F_{CG}(A)$ and $a \in A$ ($\subset F_{CG}(A)$). Thus in particular, $\tilde{\rho}(a^{-1}) = -\tilde{\rho}(a) = -\rho(a)$. The map $\tilde{(.)}$ is related to the map $\hat{(.)}$ used in the preceding section by the formula

Observation 4.4 *$\hat{\rho}(u) = \tilde{\rho}(ev(u))$, where ev (the evaluation map mentioned in Def.4.3) takes a word $u \in (A + A^{-1})^*$ (i.e. a sequence of actions and inverse actions) to the vector whose component in a is the difference between the respective numbers of occurrences of a and a^{-1} in u: $ev(u)(a) = \sharp_u(a) - \sharp_u(a^{-1})$.*

4.2 Regions in Accessible Graphs with Deterministic Labeling

An abstract region in an accessible graph represents a resource, measured at each node by a displacement (w.r.t. the initial amount) which must be the same through any initial path leading to that node.

Definition 4.5 (Regions in accessible graphs with deterministic labeling) *A* region *of G is a region of $L(G)$ which gives identical value to all labels of initial paths having the same terminal node, i.e. $\rho \in R_{L(G)}$, and for every initial paths p_1, p_2, $[\partial^1(p_1) = \partial^1(p_2) \Rightarrow \hat{\rho}(l(p_1)) = \hat{\rho}(l(p_2))$. Let R_G, resp. BR_G denote the set of regions, resp. bounded regions of G.*

Again, R_G is a monoid, BR_G is an abelian group, and *concrete* regions $\rho : S \to \mathbb{N}$ may be derived from *abstract* regions $\rho : A \to \mathbb{Z}$, where $G = [S, T, A, l]$, by assigning to each node $s \in S$, reached by some initial path p_s, the value $M_G(\rho) + \tilde{\rho}(\pi(p_s))$ $(= M_G(\rho) + \hat{\rho}(l(p_s))$ by Obs. 4.4) where $M_G(\rho) = M_{L(G)}(\rho)$. Likewise, nets derived from graphs are defined by a straightforward adaptation of Def.3.4.

Definition 4.6 (Nets derived from graphs)

1. *The* saturated net $\mathcal{N}(G)$ *derived from G is the net (R_G, A, W, M_G), where $W(\rho, a) = \rho(a)$ for $\rho \in R_G$, and $M_G(\rho) = -\inf\{\hat{\rho}(u) \mid u \in L(G)\}$.*

2. *For any set of regions $R \subseteq R_G$, the* R-net *derived from G is the sub-net of $\mathcal{N}(G)$ with set of places R.*

3. *In particular, the* saturated bounded net *$\mathcal{BN}(G)$ derived from G is the sub-net of $\mathcal{N}(G)$ with set of places BR_G.*

However, it does *not* suffice, for obtaining the necessary and sufficient conditions under which the marking graphs of nets $\mathcal{N}(G)$ or $\mathcal{BN}(G)$ are isomorphic to the given graph G, to translate literally to graph terminology the properties of separation defined earlier for languages, because isomorphism of graphs is stronger than equality of languages. The adequate properties for graphs are the following.

Definition 4.7 (Separation properties for graphs) *Let $G = [S, T, A, l]$. Two nodes $s, s' \in S$ are* separated *by a region $\rho \in R_G$ if $\hat{\rho}(u) \neq \hat{\rho}(u')$ for some words u, u' labeling initial paths with terminal nodes s, s'. Let $R \subseteq R_G$ be a subset of regions of G. Graph G is* separated by R *if $L(G)$ is separated by R and every non identical pair $s, s' \in S$ is separated by some $\rho \in R$. Graph G is* separated *if it is separated by R_G, and* boundedly separated *if it is separated by BR_G.*

Since the places of a net may always be identified with regions of its marking graph, the marking graph of a net is necessarily separated. The converse is stated in Prop. 4.9 and its corollary. In order to prove this result, we first notice the following lemma in which $\mathcal{N}_R(G)$ denotes the R-net derived from G.

Lemma 4.8 $L(G) \subseteq \mathcal{L}(\mathcal{N}_R(G))$.

Proof: By induction on the length of $u \in L(G)$. If $|u| = 0$, then $u = \epsilon$ and $\epsilon \in \mathcal{L}(\mathcal{N}_R(G))$ by definition. Take $u \cdot a \in L(G)$ –whence $u \in L(G)$– such that $|u| = n$. By induction hypothesis, $u \in \mathcal{L}(\mathcal{N}_R(G))$. Let M be the marking reached after the execution of u; for each $\rho \in R$ we have $M(\rho) = M_G(\rho) + \hat{\rho}(u)$. From $u \cdot a \in L(G)$ it follows $M_G(\rho) + \hat{\rho}(u) + \rho(a) \geq 0$ for each $\rho \in R$, hence a is enabled at M, and $u \cdot a \in \mathcal{L}(\mathcal{N}_R(G))$. ∎

Proposition 4.9 *An accessible graph with deterministic labeling is isomorphic to the derived R-net if, and only if, it is separated by R.*

Proof: ($\Rightarrow$) Suppose G is not separated by R. Then, either $L(G)$ is not separated by R, or there exist two distinct nodes s and s', which are not separated by any region $\rho \in R$. In the first case, there exists $u \cdot a \in (L(G) \cdot A) \setminus L(G)$ such that $M_G(\rho) + \hat{\rho}(u \cdot a) \geq 0$ for each $\rho \in R$. Then, by the above lemma, $u \in \mathcal{L}(\mathcal{N}_R(G))$, hence also $u \cdot a \in \mathcal{L}(\mathcal{N}_R(G))$ and G is not isomorphic to the marking graph of $\mathcal{N}_R(G)$. In the second case, take two words, u and u', labeling initial paths terminating respectively at s and s'. By hypothesis, for each $\rho \in R$, $\hat{\rho}(u) = \hat{\rho}(u')$, whence $M = M'$ where M and M' are the markings reached after executing respectively u and u', and G is not isomorphic to the marking graph of $\mathcal{N}_R(G)$ since $s \neq s'$
($\Leftarrow$) Suppose G is separated. To each node s of G, let us associate the map $\gamma_s : R \to I\!N$ defined as follows: $\forall \rho \in R\ \gamma_{s_0}(\rho) = M_G(\rho)$, where s_0 is the initial node of G and $\gamma_s(\rho) = M_G(\rho) + \hat{\rho}(u)$, where u labels an (arbitrary) initial path terminating in s. So γ_s is a potential marking of $\mathcal{N}_R(G)$; in particular, γ_{s_0} coincides with the initial marking. Notice that, $\gamma_s \neq \gamma_{s'}$ for $s \neq s'$ since G is separated. Suppose now that action a is enabled at marking γ_s. This implies $M_G(\rho) + \hat{\rho}(u) + \rho(a) \geq 0$ for each $\rho \in R$, where u labels an initial path terminating in s, whence $u \cdot a \in L(G)$, since $L(G)$ is separated by R. Therefore, by determinacy of G, there exists a node s' such that $s \xrightarrow{a} s'$. By definition of the mapping γ, $\gamma_s[a > \gamma_{s'}$. Hence, every reachable marking is of the form γ_s for some $s \in S$, and the marking graph of $\mathcal{N}_R(G)$ is isomorphic to G because the latter is an accessible graph. ∎

Corollary 4.10 *Given an accessible graph G with deterministic labeling:*

1. *G is isomorphic to the marking graph of the saturated net $N(G)$ iff it is separated;*

2. *G is isomorphic to the marking graph of the saturated bounded net $BN(G)$ iff it is boundedly separated.*

Observing that every region in a finite graph is finite, we concentrate in the rest of the section on the problem of synthesizing finite nets from finite graphs.

4.3 Regions in Finite Accessible Graphs with Deterministic Labeling

From now on, $G = [S, T, A, l]$ is a *finite* accessible graph with deterministic labeling on $A = \{a_1, \ldots, a_n\}$. Therefore, $R_G = BR_G$, and this module is a sub-module of the finite dimensional $\mathbb{Z}$-module $BR_{L(G)}$. In a first stage, we will provide an algorithm which computes in polynomial time a basis for the module of regions R_G. Let us start with an algebraic characterization of regions.

Proposition 4.11

1. *A map $\rho : A \to \mathbb{Z}$ is a bounded region of the* language $L(G)$ *iff $\tilde{\rho}(\pi(c)) = 0$ for every directed cycle c in G.*

2. *A map $\rho : A \to \mathbb{Z}$ is a bounded region of graph G iff $\tilde{\rho}(\pi(c)) = 0$ for every cycle c in G.*

   Proof: The first assertion is immediate from Obs. 4.4 which tells us that $\tilde{\rho}(\pi(c)) = \hat{\rho}(l(c))$ for every directed cycle c in G. *A fortiori*, considering now the second assertion, any map $\rho : A \to \mathbb{Z}$ such that $\tilde{\rho}(\pi(c)) = 0$ for every cycle c in G is a bounded region of the *language* $L(G)$. Moreover, a map ρ as above must be a bounded region of G because $p_1 \cdot p_2^{-1}$ is a cycle whenever two initial paths p_1 and p_2 have the same terminal node, which entails $[\hat{\rho}(l(p_1)) = \tilde{\rho}(\pi(p_1)) = \tilde{\rho}(\pi(p_2)) = \hat{\rho}(l(p_2))]$. Conversely, let ρ be a bounded region of G, and let $c = (t_0, \ldots, t_{k-1})$ be a cycle (i.e. $t_i \in T + T^{-1}$). Since G is an accessible graph, one can find initial paths p_i with $\partial^1(p_i) = \partial^0(t_i)$; since ρ is a bounded region of G, $\tilde{\rho}(\pi(t_i)) = \tilde{\rho}(\pi(p_{(i+1) \bmod k})) - \tilde{\rho}(\pi(p_i))$; thus $\tilde{\rho}(\pi(c)) = \tilde{\rho}(\pi(t_0)) + \ldots + \tilde{\rho}(\pi(t_{k-1})) = 0$. ∎

Proposition 4.11 tells us that R_G, the module of regions of G represented as n-vectors $< \rho(a_1), ..., \rho(a_n) >$, is the kernel of a linear transformation from $\mathbb{Z}^n$ to $\mathbb{Z}^\alpha$ for α equal to the cardinal of the set of cycles in G. Surely, α may be infinite, but there are at most n linearly independent constraints $\vec{\pi}(c) \cdot \vec{\rho} = 0$ imposed by cycles c on regions $\vec{\rho}$. According to a usual practice in graph theory (see e.g. [Ber70]), let us represent a cycle c as a vector $\vec{c} \in (T \to \mathbb{Z})$. It is a well known fact that all vectors representing cycles in G may be generated from

a basis of $\nu(G)$ linearly independent vectors, with $\nu(G) = |T| - |S| + 1$. Let us observe that π is a linear operator from $F_{CG}(T)$ $(= (T \to \mathbb{Z}))$ to $F_{CG}(A)$ $(= (A \to \mathbb{Z}))$. Therefore, $\nu(G)$ equations of the form $\vec{\pi}(c) \cdot \vec{\rho} = 0$ suffice to generate all of them! Summing up, the following proposition holds.

Proposition 4.12 *Let $\{\vec{c}_1 \ldots \vec{c}_{\nu(G)}\}$ be a basis of cycles of $G = [S, T, A, l]$, where $T = \{t_1 \ldots t_m\}$ and $A = \{a_1 \ldots a_n\}$, then R_G is the kernel of the linear transformation defined by the $(\nu(G) \times n)$ matrix $\vec{M}_G$ with integral elements:*

$$M_G(i,j) = \sum\{\vec{c}_i(t_k) \mid (1 \leq k \leq m) \wedge (l(t_k) = a_j)\}.$$

If we can compute the integral matrix M_G, the algorithm of von zur Gathen and Sieveking yields now in polynomial time a basis of linearly independent vectors $\{\vec{\rho}_1 \ldots \vec{\rho}_t\}$ for the module of regions R_G.

4.4 Computing a Spanning Tree and a Basis of Cycles for G

We show here how computing (in polynomial time) a basis of cycles for a finite and accessible graph $G = [S, T]$. In view of the following proposition, borrowed from [GM85], this task reduces to constructing a spanning tree in graph G.

Proposition 4.13 (Gondran and Minoux) *Let $G = [S, T]$ be a finite graph with p connected components. Let $\mathcal{F} = [S, U]$ be a maximal* forest *(i.e. graph without cycle) in G. For $t \in (T - U)$, let c^t be the cycle with set of arcs $U + \{t\}$ (this cycle is unique up to reversal). Cycles c^t, for t ranging over $(T - U)$, form a basis of cycles of G, with dimension $\nu(G) = |T| - |S| + p$.*

In case when G is an accessible graph, the maximal forest of the proposition may naturally be chosen among the spanning trees rooted at the initial node of the graph. Constructing a spanning tree takes polynomial time, hence computing a basis of cycles for G takes polynomial time.

4.5 Deciding upon Separatedness of a Finite Accessible Graph

Let $G = [S, T, A, l]$ be a finite accessible graph, and let $Span(G)$ be a spanning tree of G. By Def. 4.7, graph G is separated if and only if the following problems are solvable:

Problem 4.14 *For each non identical pair of initial paths p, p' in $Span(G)$, including the empty path ε, find a linear combination $\vec{\rho} = \lambda_1 \vec{\rho}_1 + \ldots + \lambda_t \vec{\rho}_t$ of the base vectors of the module R_G such that $\tilde{\rho}(\pi(p)) \neq \tilde{\rho}(\pi(p'))$.*

Problem 4.15 *For each action $a \in A$ and for each initial path p in $Span(G)$ such that $(l(p) \cdot a) \notin L(G)$, where possibly $p = \varepsilon$, find a linear combination $\vec{\rho} = \lambda_1 \vec{\rho}_1 + \ldots + \lambda_t \vec{\rho}_t$ of the base vectors of the module R_G such that $\tilde{\rho}(\pi(p)) + \rho(a) - \tilde{\rho}(\pi(p')) < 0$ for every path p' in $Span(G)$.*

Now Prob. 4.14 is trivial, whereas Prob. 4.15 may be solved in polynomial time following Khachiyan's method (along the same lines as in section 3.3). Hence, we can sum up the section as follows.

Theorem 4.16 *Let G be a finite and accessible graph with deterministic labeling, then one may decide whether G is isomorphic to the case graph of some finite (and irreducible) net, and construct that net, in polynomial time.*

5 Conclusion

In the introduction the notion of synchronic distance is briefly mentioned; it was introduced, in the context of net theory, by C.A. Petri as a tool to measure the relative degree of freedom between sets of transitions in a concurrent system. The close relation between synchronic distances and regions has been discussed in some detail in [BDP93] –where further references can be found–. In brief, to each region in a separated graph corresponds a finite synchronic distance and vice versa. Actually, the existence of a bounded abstract region in a language or in a graph implies a constraint on the relative frequence of execution of the transitions affecting the region itself. The range of variation

of the region, that is, the difference between its minimum and maximum values, is a measure of the reciprocal independence: higher values mean looser constraints. This is in agreement with the interpretation of regions as abstract resources shared by transitions, given in this paper. Hence, the algorithms presented here can be seen as a way to compute efficiently synchronic distances. A slightly different approach to synchronic relations in Petri nets was set out in [SC88], where linear programming techniques are used to compute upper bounds of so-called synchronic invariants in a given net.

We should finally explain why the algorithmic methods which have been proposed in this paper cannot, in our opinion, be adapted to the synthesis of condition-event nets or elementary nets. In the notation of section 4, a region ρ in a graph G may be identified with a condition iff $|\tilde{\rho}(\pi(p)) - \tilde{\rho}(\pi(p'))| \leq 1$ for any pair p, p' of initial paths in the spanning tree of G. With the synthesis of condition-event nets in view, that constraint on regions must be accounted for in a stronger statement of Prob. 4.15. As a result, one obtains a linear system $M\vec{\Lambda} \leq (-1)^H, N\vec{\Lambda} \geq (-1)^H$ with unknown the integral vector $\vec{\Lambda}$, where M and N are integral matrices. That system may still be solved in the rational field, but we can no more derive integral solutions from rational solutions by arbitrary multiplications, since the two comparisons are in opposite directions. This leaves no hope to adapt the algorithms to condition-event nets.

References

[Ber70] Berge, C., *Graphes et hypergraphes.* Dunod, Paris (1970).

[Ber93] Bernardinello, L., *Synthesis of Net Systems.* Application and Theory of Petri Nets, Springer-Verlag Lecture Notes in Computer Science, vol. 691 (1993) 89–105.

[BDP93] Bernardinello, L., De Michelis, G., and Petruni, K., *Synchronic distances as Generalized Regions.* Rapporto Interno n. 107–93, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano (1993).

[DR92] Desel, J., and Reisig, W., *The Synthesis Problem of Petri Nets.* TUM research report, Munich (1992).

[DS92] Droste, M., and Shortt, R.M., *Bounded Petri Nets of Finite Dimension have only Finitely Many Reachable Markings.* Bulletin of the European Association for Computer Science, number 48, (1992) 172–174.

[DS93] Droste, M., and Shortt, R.M., *Petri Nets and Automata with Concurrency Relations – an Adjunction.* in "Semantics of Programming Languages and Model Theory", M. Droste and Y. Gurevich eds(1993) 69–87.

[ER90] Ehrenfeucht, A., and Rozenberg, G., *Partial 2-structures ;* Part I : *Basic Notions and the Representation Problem*, and Part II : *State Spaces of Concurrent Systems*, Acta Informatica, vol 27 (1990).

[GM85] Gondran, M., and Minoux, M., *Graphes et algorithmes.* Eyrolles, Paris (1985).

[HKT92] Hoogers, P.W., Kleijn, H.C.M., and Thiagarajan, P.S., *A trace semantics for Petri nets.* Springer-Verlag, Lecture Notes in Computer Science, vol. 623 (1992) 595–604.

[Muk93] Mukund, M., *Petri Nets and Step Transition Systems.* International Journal of Foundation of Computer Science, vol 3, n^o 3 (1993).

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

[SC88] Silva, M. and Colom, J.M., *On the computation of structural synchronic invariants in P/T nets.* In "Advances in Petri Nets 1988", G. Rozenberg (Ed.), Springer-Verlag, Lecture Notes in Computer Science, vol. 340 (1988) 386–417.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399