Linear Time Algorithms for Liveness and Boundedness in Conflict-free Petri Nets*

Paola Alimonti† Esteban Feuerstain†‡

Umberto Nanni§

TR-92-008

February 1992

Abstract

In this paper we consider the problems of deciding the set of potentially firable transitions, the liveness and boundedness for the class of *Conflict-Free Petri Nets*. For these problems we propose algorithms which are linear in the size of the description of the net, dramatically improving the best previous known results for these problems [11, 10]. Moreover the algorithm for the first problem is *incremental*: it is possible to perform an arbitrary sequence of *updates*, introducing new transitions and increasing the initial marking of the net, and *queries*, asking whether any transition is firable or any place reachable. Queries are answered in constant time, and the total cost for all the modifications is still linear in the size of the final net. Our approach is based on a representation of conflict-free Petri nets by means of *directed hypergraphs*.

*Work supported by the ESPRIT II Basic Research Action Program of the European Community under contract No.3075 (Project ALCOM) and by the Italian Project "Algoritmi e Strutture di Calcolo", Ministero dell'Università e della Ricerca Scientifica e Tecnologica.

†Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, I-00198 Roma, Italia.

‡On leave from ESLAI (Escuela Superior Latinoamericana de Informática), partially supported by a grant from Fundación Antorchas, Argentina.

§Dipartimento di Matematica Pura e Applicata, University of L'Aquila, via Vetoio, Coppito - 67010 - L'Aquila, Italy; and International Computer Science Institute, 1947 Center St., Berkeley, CA 94704, U.S.A.

1 Introduction

Petri nets [19, 20, 18] have been widely used to design and modeling concurrent systems, as well as certain kinds of time-dependent systems. This is due not only to their neat graphical representation but also to the fact that a great quantity of theoretical studies have been done on their mathematical properties. These properties reflect the properties and patterns of behaviour of the systems being modeled. As a modeling tool, and specially as a design tool, it would be desirable to have efficient algorithms that allow us to decide if a system satisfies a certain property. Unfortunately, this is not generally the case, and research in this field has been somehow disappointing, since, on one hand, some of these properties have been proved to be decidable, and on the other hand, from the algorithmical point of view, the great majority of them has proved to be extremely difficult, if not definitively intractable.

It is the case, however, that for some subclasses of Petri nets, this kind of problems can be solved more efficiently than for arbitrary nets (see, e.g., [6, 8, 9, 10, 11, 13, 14, 15, 17]). One of the simplest of such classes is that of *conflict-free* Petri nets [10, 11, 17]. This class of nets is clearly less powerful than the class of general nets, as it imposes a restrictive use of an important feature such as nondeterminism. Nevertheless, it can still be used to model certain kinds of distributed systems. Besides, conflict-free Petri nets are equivalent to the controls of decision-free flow-chart schemata studied in [15].

The problem of deciding whether a transition is potentially firable or not and liveness and boundedness problems, that require at least exponential time for arbitrary nets, have been shown to be polynomially solvable for conflict-free nets [10, 11]. More precisely, in those works are presented algorithms for solving these problems for conflict-free Vector Replacement Systems (VRS) [6, 11], a different formalism that includes as a special case that of conflict-free Petri nets. The algorithms presented in that work require $O(n^{1.5})$ time for VRS and $O(n^2)$ time for Petri nets, where n is the size of the VRS or the net respectively. This difference is due to the fact that, while VRS are described necessarily by matrices, a Petri net can be described more succinctly by, for example, adjacency lists.

In this paper, in order to capture some interesting structural properties of conflict-free Petri nets we consider *directed hypergraph* [1]. This representation allows us to obtain very efficient algorithms and data structures to handle conflict-free Petri nets. More precisely, we propose linear time algorithms for determining potentially firable and live transitions and boundedness of the net. Moreover, we maintain the set of potentially firable transitions *on-line*, that is, while the net is modified: an important feature while dealing with the interactive design of a system. In other words, our algorithms and data structures update the set of potentially firable transitions (and reachable places) during the incremental construction of the net. The overall cost of all these updates is linear in the description of the net.

The reminder of this paper is organized as follows: Section 2 states the basic definitions and notations. Section 3 deals with the representation of Petri nets as hypergraphs and the determination of the potential firability of transitions. In sections 4 and 5 we describe properties and algorithms related to the liveness and boundedness problems for conflict-free Petri nets. In section 6 we describe conclusions and open problems. The implementation of the incremental algorithm for the first problem is described in the Appendix.

2 Basic definitions and notation

A *Petri net* is a 4-tuple $\mathcal{P} =< P, T, A, M_0 >$, where P is a finite set of *places*; T is a finite set of *transitions*; A is a finite set of *arcs*, $A \subseteq (P \times T) \cup (T \times P)$; and $M_0 : P \to N$, N the set of natural numbers, is the *initial marking*. Note that we consider Petri nets without weighted arcs.

Let t be a transition in T, ${}^o t = \{p|(p,t) \in A\}$, and $t^o = \{p|(t,p) \in A\}$, are called the *input set* and the *output set* of t. The notation is extended to the places.

A transition t is said to be *enabled* when each input set of t contains at least one token. We write $M \xrightarrow{t} M'$ ($M \xrightarrow{t}$) to indicate that t is enabled in the marking M and the firing of t yields the marking M' (t is enabled in the marking M). The notation is extended to a sequence of transitions, $\sigma \in T^*$, called *firing sequence*.

A transition t (or a sequence σ) is said to be *potentially firable* if there exists some sequence σ' such that $M_0 \xrightarrow{\sigma'} M \xrightarrow{t}$ ($M_0 \xrightarrow{\sigma'} M \xrightarrow{\sigma}$) otherwise it is said to be *dead*.

A place p is *reachable* if there exists a firable sequence σ such that $M_0 \xrightarrow{\sigma} M$ and $M(p) \neq 0$.

The *set of reachable markings* or the *reachability set* of the Petri net $\mathcal{P} =< P, T, A, M_0 >$, is $R(M_0) = \{M | M_0 \xrightarrow{\sigma} M$, for some $\sigma \in T^*\}$. The notation is extended to every marking $M \in R(M_0)$.

A transition $t \in T$ is said to be *live* if, for any $M \in R(M_0)$, there is a marking $M' \in R(M)$ such that $M' \xrightarrow{t}$. A Petri net $\mathcal{P} =< P, T, A, M_0 >$ is said to be *live* if every transition $t \in T$ is live in $\mathcal{P}$. Note that a live transition is firable infinitely many times from any marking $M \in R(M_0)$.

A place $p \in P$ is said to be *bounded*, if there exists a constant k such that, for any $M \in R(M_0)$, $M(p) < k$. A Petri net $\mathcal{P} =< P, T, A, M_0 >$ is said to be *bounded* if every place $p \in P$ is bounded.

A Petri net $\mathcal{P} =< P, T, A, M_0 >$ is *persistent* if for all $t, t' \in T$, $t \neq t'$, and any reachable marking M, $M \xrightarrow{t}$ and $M \xrightarrow{t'}$, imply $M \xrightarrow{t.t'}$, that is if the transitions t and t' are enabled in M, then the firing of one cannot disable the other.

A Petri net $\mathcal{P} =< P, T, A, M_0 >$ is *conflict-free* if each place $p \in P$ satisfies: either there is at most one arc out of p (p is said *unbranched*), or for each transition t for which $p \in {}^o t$, then $p \in t^o$ (p is said *branched*). It follows that a conflict-free Petri net is persistent for any initial marking.

Note the following basic property of conflict-free Petri nets (without multiple or weighted arcs): the set of potentially firable transitions does not change if we change the cardinality of the initial marking from M_0 to M_0' in such a way that $M_0' = 0$ if and only if $M_0 = 0$. In particular this allows to consider only initial markings with at most one token for each place.

3 Petri Nets and Hypergraphs

In this section we consider the *directed hypergraphs* [4, 5, 1] to capture significant structural properties of conflict-free Petri nets.

Directed hypergraphs have been extensively used as a suitable mathematical representation model in different areas of computer science [3, 7], such as functional dependences in relational databases [1], logic programming [2], problem solving [5].

Moreover we make use of *on-line* techniques [21], useful in the design of a conflict-free Petri net, to maintain the set of potentially firable transitions and reachable places while incrementally updating the net.

Dynamic techniques for similar problems on graphs and hypergraphs have been developed for example in [12, 16, 3].

Definition 3.1 *A* directed hypergraph $\mathcal{H}$ *is a pair* $< N, H >$ *where* N *is the set of nodes,* H *is the set of* hyperarcs *and for each* $h \in H$, $h =< X, y >$, *with* $X \subseteq N$ *and* $y \in N$.

Definition 3.2 *Let* $\mathcal{H} =< N, H >$ *be a hypergraph,* $X \subseteq N$ *be a nonempty set of nodes and* y *be a single node in* N. *A* hyperpath *from* X *to* y *is a (possibly empty) set* $P_{X,y}$ *of hyperarcs such that* $P_{X,y} \subseteq H$ *and at least one of the following conditions holds:*

I) extended reflexivity*:* $y \in X$ *and* $P_{X,y}$ *is empty;*

II) extended transitivity*: there exists a set of nodes* $Z \subseteq N$ *such that* $< Z, y >$ *is a hyperarc in* $\mathcal{H}$, *for each* $z \in Z$ *there exists a hyperpath* $P_{X,z}$ *from* X *to* z, *and*

$$P_{X,y} = \{< Z, y >\} \cup \bigcup_{z \in Z} P_{X,z}.$$

Definition 3.3 *Let* $\mathcal{P} =< P, T, A, M_0 >$ *be a Petri net. The hypergraph associated to* $\mathcal{P}$ *is the hypergraph* $\mathcal{H}_{\mathcal{P}} =< N_P \cup N_T, H >$ *where:*

a) $N_P = P$ *is the set of* P-nodes*;*

b) $N_T = T$ *is the set of* T-nodes*;*

c) $H = \{< \{t\}, p > \mid t \in N_T, p \in N_P,$ *and* $p \in t^o\} \cup \{< {}^o t, t > \mid t \in N_T\}$ *is the set of* hyperarcs.

We will denote as $M_0 \subseteq N_P$ the set of P-nodes $M_0 = \{p | M_0(p) > 0\}$ in $\mathcal{P}$.

In figure 1 a conflict-free Petri net and the associated hypergraph are shown.

Hyperpaths in $\mathcal{H}_{\mathcal{P}}$ correspond to firable sequences in a conflict-free Petri net, as the following theorem proves.

Theorem 3.1 *Let* $\mathcal{P} =< P, T, A, M_0 >$ *be a conflict-free Petri net and* $\mathcal{H}_{\mathcal{P}}$ *be the associated hypergraph.*

1. *If there exists a hyperpath from* M_0 *to the T-node* t *in* $\mathcal{H}_{\mathcal{P}}$, *then the transition* t *is potentially firable in* $\mathcal{P}$.

2. *If there exists a hyperpath from* M_0 *to the P-node* p *in* $\mathcal{H}_{\mathcal{P}}$, *then the place* p *is reachable in* $\mathcal{P}$.

Proof. By induction on the cardinality of the hyperpath.

Basis.

1. $|P_{M_0,t}| = 1$. This means that the hyperpath consists only of a single hyperarc $< {}^{o}t, t >$, where ${}^{o}t \subseteq M_0$. Thus t is potentially firable in $\mathcal{P}$.

2. $|P_{M_0,p}| = 0$. Since this means that $p \in M_0$ then p is reachable in $\mathcal{P}$.

Inductive step.

1. Let $P_{M_0,t}$ be a hyperpath whose cardinality is n. Since, by definition 3.2, there exists a hyperarc $< {}^{o}t, t > \in P_{M_0,t}$ and moreover for each $p \in {}^{o}t$ there exists a (possibly empty) hyperpath $P_{M_0,p} \subseteq P_{M_0,t}$ whose cardinality is less than n, then by inductive hypothesis for any $p \in {}^{o}t$, p is reachable in $\mathcal{P}$; in a conflict-free Petri net this implies that the transition t is potentially firable.

2. Analogously, due to the existence of a hyperpath $P_{M_0,p}$ in $\mathcal{H}_{\mathcal{P}}$, there exists a hyperarc $< t, p > \in P_{M_0,p}$ and there exists a hyperpath $P_{M_0,t} \subseteq P_{M_0,p}$ whose size is less than n; this means that t is potentially firable in $\mathcal{P}$and then p is reachable.

□

On the other side, the existence of a hyperpath from M_0 to a T-node t is a necessary condition for the potential firability of the transition t in $\mathcal{P}$ also for general (i.e. not necessarily conflict-free) Petri nets.

Theorem 3.2 *Let $\mathcal{P} =< P, T, A, M_0 >$ be a general Petri net and $\mathcal{H}_{\mathcal{P}}$ be the associated hypergraph.*

1. *If a transition t is potentially firable in $\mathcal{P}$ then there exists a hyperpath from M_0 to the T-node t in $\mathcal{H}_{\mathcal{P}}$.*

2. *If a place p is reachable in $\mathcal{P}$ then there exists a hyperpath from M_0 to the P-node p in $\mathcal{H}_{\mathcal{P}}$.*

Proof. We will prove both the above points (1) and (2) considering a firable sequence $\sigma =< t_1, t_2, \ldots, t_n >$ which either enables t or allows to reach p. We will prove that the arcs used in σ build up a hyperpath P_{M_0,t_n}. The case (2) will follow from the consideration that if t_j is such that $p \in t_j^o$ (with $j = 1, 2, \ldots, n$), then, according definition 3.2, the hyperpath from M_0 to p in $\mathcal{H}_{\mathcal{P}}$ can be built in the following way:

$$P_{M_0,p} =< \{t_j\}, p > \cup P_{M_0,t_j}.$$

We proceed by induction on the length of the sequence σ.

Basis: $|\sigma| = 1$.

Since $\sigma =< t_1 >$ is a firable sequence in $\mathcal{P}$, then ${}^{o}t_1 \subseteq M_0$ and by definition there exists a hyperpath $P_{M_0,t_1} = \{<^{o} t_1, t_1 >\}$.

Inductive step: $\sigma =< t_1, t_2, \ldots, t_n >$ is firable and for each shorter sequence the theorem is true.

Let $\sigma_i =< t_1, t_2, \ldots, t_i >$ be the firable subsequence containing the first i elements of σ, and σ_0 be the empty sequence. For each positive $i < n$, we have $M_0 \xrightarrow{\sigma_{i-1}} M_{i-1} \xrightarrow{t_i} M_i$, and, by inductive hypothesis, there exists a hyperpath from M_0 to t_i.

For each $p \in M_{n-1}$ we have that there exists a hyperpath $P_{M_0,p}$ from from M_0 to p. In fact either $p \in M_0$ (and then there exists an empty hyperpath from M_0 to p) or $p \in t_j^o$ for some $j < n$. In the latter case, being the sequence σ_j firable, and due to the inductive hypotheses, there exists a hyperpath $P_{M_0,p} =< \{t_j\}, p > \cup P_{M_0,t_j}$.

Since σ_n is firable, for each $p \in {}^o t_n$, also $p \in M_{n-1}$ holds and we have that, by transitivity (definition 3.2), there exists a hyperpath from M_0 to t_n given by:

$$P_{M_0,t} =< {}^o t_n, t_n > \cup \bigcup_{p \in {}^o t_n} P_{M_0,p}.$$

□

We remark that, as an obvious consequence of theorems 3.1 and 3.2, we have that for conflict-free Petri nets the existence of a hyperpath from M_0 to a given transition node t in $\mathcal{H}_\mathcal{P}$ is a necessary and sufficient condition for the potential firability of the transition t in $\mathcal{P}$.

Now we can state the result proving that the incremental design of a conflict-free Petri net, while maintaining information about the potential firability of the transitions, can be performed very efficiently.

Theorem 3.3 *Let $\mathcal{P} =< P, T, A, M_0 >$ be a conflict-free Petri net, with $n_T = |T|$, $n_P = |P|$, $n_A = |A|$. There exist data structures and algorithms to perform the following operations:*

a) insert a (disconnected) place in P;

b) define a new transition t together with its input set ${}^o t$;

c) insert an arc $< t, p >$ in A;

d) add a place p to the initial marking M_0.

The total time required to perform an arbitrary sequence of operations of the above kinds, starting from an empty Petri net, is $O(n_A + n_P + n_T)$, where the cardinalities refer to the final net. Deciding whether a given transition is potentially firable requires constant time.

The algorithms (partially derived from techniques used in [12, 3]) and the proof are given in the appendix. We also show how the proposed algorithms can be used to maintain, while modifying the net, a copy of the reachable portion of the net itself. This net will be used as input to the algorithms for liveness and boundedness.

As a corollary of Theorem 3.3 we have the following obvious result about the potential firability of transition in a conflict-free Petri net.

Corollary 3.4 *The problem of finding the set of potential firable transitions $T_F \subseteq T$ in a conflict-free Petri net is $O(n_A + n_P + n_T)$, that is linear in the description of the net.*

4 Liveness

In this section we examine the liveness problem for conflict-free Petri nets and propose a linear algorithm that determines the set of live transitions of a given conflict-free Petri net. The concept of liveness is tied to that of deadlock and deadlock-freeness, as if a transition is not live, then it is possible to reach a marking in which the transition is not potentially firable, perhaps signifying a possible deadlock. Thus, when modeling a distributed system, it may be important to decide whether a transition is live or not (see [18]).

The following lemmas show important proprieties of conflict-free Petri nets related to the liveness problem.

Lemma 4.1 *Let $\mathcal{P} =< P, T, A, M_0 >$ be a conflict-free Petri net. If a transition t is potentially firable in M_0, then it is also potentially firable in M, where $M_0 \xrightarrow{t'} M$ and $t' \neq t$.*

Proof. If t is potentially firable in the marking M_0, then there is a sequence σ enabled in M_0, such that $M_0 \xrightarrow{\sigma} M \xrightarrow{t}$. Let t' be a transition enabled in M_0, and M' such that $M_0 \xrightarrow{t'} M'$. Suppose that σ is not firable in M'. Therefore there must be a place $p \in {}^{o}t'$ such that $p \notin t'^{o}$ and $p \in {}^{o}t''$, with $t'' \in \sigma$, that is the transition t' takes without putting back a token from at least one place in the input of a transition used in σ. Since the net is conflict-free this transition must be t'. Then the transition t' is used in the sequence σ. Hence we can write $\sigma = \sigma_1.t'.\sigma_2$, and $M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t'} M_2 \xrightarrow{\sigma_2} M$, where t' is not used in σ_1. Now suppose that the sequence σ_1 cannot fire after t'. Let t'' be the first transition of σ_1 which cannot fire, because t' took a token from at least one of its input places. This is a contradiction since t' is not used in σ_1 and the net is conflict-free. So, σ_1 can be executed after t' and we can write $M_0 \xrightarrow{t'} M' \xrightarrow{\sigma_1.\sigma_2} M \xrightarrow{t}$, that is t may still fire. Otherwise, if t' is not used in σ, this sequence can fire after t', so we can write $M_0 \xrightarrow{t'} M' \xrightarrow{\sigma} M'' \xrightarrow{t}$. □

Corollary 4.2 *Let $\mathcal{P} =< P, T, A, M_0 >$ be a conflict-free Petri net, if a transition t is potentially firable in M_0, then it is also potentially firable in M, with $M_0 \xrightarrow{\sigma} M$, and t is not used in σ.*

Corollary 4.3 *Let $\mathcal{P} =< P, T, A, M_0 >$ be a conflict-free Petri net, then there exists a sequence σ firable in M_0 which uses every potentially firable transition.*

Lemma 4.4 *Given a conflict-free Petri net $\mathcal{P} =< P, T, A, M_0 >$, let $C \subseteq T$ be a subset of transitions of $\mathcal{P}$, such that $(\forall t \in C)$ t is potentially firable in M_0, and $\bigcup_{t \in C} {}^{o}t \subseteq \bigcup_{t \in C} t^{o}$; let σ be a firable sequence which uses every transition in C; and let $t_{\sigma,C}$ be the first transition in C which occurs in σ for the last time. Then $t_{\sigma,C}$ is firable after σ.*

Proof. Let $p \in {}^{o}t_{\sigma,C}$, therefore after the last firing of $t_{\sigma,C}$ in σ, at least a transition t' will fire for which $p \in t'^{o}$, because all transitions in C will fire and $p \in \bigcup_{t \in C} t^{o}$. In fact, since the net is conflict-free, if for some transition $t'' \neq t_{\sigma,C}$ $p \in {}^{o}t''$ then $p \in t''^{o}$. Hence $t_{\sigma,C}$ is enabled after σ. □

Theorem 4.5 *Given a conflict-free Petri net* $\mathcal{P} =< P, T, A, M_0 >$*, a transition* $t \in T$ *is firable infinitely many times if and only if there exists a set of transitions* $C \subseteq T$ *such that*

- $t \in C$,
- $(\forall t' \in C)$ t' *is potentially firable in* M_0,
- $\bigcup_{t' \in C} {}^{o}t' \subseteq \bigcup_{t' \in C} t'^{o}$.

Proof.

($\Leftarrow$) From Corollary 4.3, there exists a sequence σ firable in M_0 which uses at least once every transition in C. Consider the following sequence of firable sequences

$$\sigma_0 = \sigma$$

$$\sigma_{i+1} = \sigma_i . t_{\sigma_i, C}$$

with $t_{\sigma_i,C}$ defined as before. From Lemma 4.4, every σ_i is firable. Besides, also the infinite sequence $\sigma_\infty = \lim_{i \to \infty} \sigma_i$ is firable, and $(\forall t \in C)$ t occurs infinitely often in σ_∞.

($\Rightarrow$) If $t \in T$ is firable infinitely often then there exist an infinite sequence σ such that it uses t infinitely often. Let $C = \{t \in \sigma \text{ such that } |t|_\sigma = \infty\}$, where $|t|_\sigma$ is the number of occurrences of t in σ. Therefore $(\forall t \in C)$, t occurs infinitely often in σ. We have to prove that $\bigcup_{t \in C} {}^{o}t \subseteq \bigcup_{t \in C} t^{o}$. Let $p \in {}^{o}t'$ with $t' \in C$. As t' fires infinitely often, there exists a transition t'', such that $p \in t''^{o}$, which fires infinitely many times in σ. Then $t'' \in C$, that is $p \in \bigcup_{t \in C} t^{o}$.

□

In the reminder of this paper, and when no confusion arises, the conditions referred in Theorem 4.5 will be referred to as *liveness conditions*, both speaking about a set C or a transitions t in a set satisfying them.

With these results we propose the following algorithm, which determines the set of live transitions of a given conflict-free Petri net. It is intended to work over the subnet of potentially firable transitions $\mathcal{P}_F$ that can be built in linear time from the set T_F produced by the algorithm in the previous section.

Algorithm `liveness`

```
begin
   for each p ∈ P_F let count(p) ← |°p|;
   T_L ← T_F;
   S ← {p ∈ P_F | count(p) = 0};
   while S not empty
   begin
      choose p ∈ S;
      delete p from S;
      for each t ∈ p°
      begin
```

```
            delete t from T_L;
            for each p' ∈ t^o
            begin
               decrement count(p');
               if count(p') = 0 then insert p' in S
            end
         end
      end
end.
```

Theorem 4.6 *The previous algorithm determines in linear time the set of live transitions of a given conflict-free Petri Net.*

Proof. We will show that after the execution of the algorithm the set T_L contains exactly the live transitions. For this, we will use the characterization of live transitions given by Theorem 4.5. We will prove:$\forall t\ ((\exists C | \bigcup_{t\in C} {}^{o}t \subseteq \bigcup_{t\in C} t^{o}) \Leftrightarrow t \in T_L)$, that is a transition is live iff it belongs to the set T_L determined by the algorithm.

($\Rightarrow$) After the initialization, all potentially firable transitions are part of T_L, and the algorithm proceeds by deleting transitions in each step. Consider the first transition t that is deleted such that satisfies the liveness conditions. Then there is a place $p \in {}^{o}t$ such that at a certain moment $count(p) = 0$. But then no transition t' such that $t' \in {}^{o}p$ is part of T_L, and hence each one of those t' must have been deleted from T_L before. But liveness conditions state that at least one of these t' with $p \in t'^{o}$ is in the same C as t, and t was the first transition satisfying the liveness condition to be deleted, a contradiction.

($\Leftarrow$) Consider a transition t such that $t \in T_L$, then every $p \in {}^{o}t$ has $count(p) \neq 0$, then for every one of these p there is a transition $t_p \in T_L$ such that $p \in t_p^{o}$, then T_L fulfills the liveness conditions.

As far as the running time of the algorithm is concerned, it is easy to see that the algorithm works in linear time: the initialization requires no more than one visit to the net and the "while" loop is executed at most n_P times, each edge being considered at most once for all the executions of the algorithm.

□

Note: The previous algorithm can be easily modified to obtain the graph $\mathcal{G}_L$ that represents the live subnet $\mathcal{P}_L =< P_L, T_L, A_L, M_{0_L} >$ of $\mathcal{P}$.

5 Boundedness

Another important property of Petri nets is boundedness. If a system modeled by a Petri net is to be implemented, then the Petri net must be bounded, as the capacity of any given hardware is limited . In order to develop an algorithm for the boundedness problem for conflict-free Petri Nets, we will show necessary and sufficient conditions that the set T_L determined by the algorithm in Section 4 has to fulfill. In [15] Karp and Miller showed that a Petri net is unbounded if and only if it can execute a positive loop. That is, there exists

a set $C \subseteq T$ such that the firing of a sequence σ which uses exactly once every transition in C, $M \xrightarrow{\sigma} M'$ such that $M' > M$.

Lemma 5.1 *A conflict free Petri net $\mathcal{P} =< P, T, A, M_0 >$ is bounded iff for any set $C \subseteq T$ satisfying the liveness conditions, we have $\sum_{t \in C} |{}^o t| = \sum_{t \in C} |t^o|$.*

Proof.

($\Rightarrow$) From Theorem 4.5 if $\bigcup_{t \in C} {}^o t \subseteq \bigcup_{t \in C} t^o$ and, for any $t \in C$, t is potentially firable in M_0, then there exists a sequence σ potentially firable such that $M \xrightarrow{\sigma} M'$, with $M' \geq M$, where $(\forall t \in C)$ t occurs exactly once in σ. Since the net is bounded, we must have $M' = M + \delta_\sigma = M$, where δ_σ is the displacement of σ. Therefore $\delta_\sigma = 0$, that is, the quantity of tokens consumed by the firing of the transitions in C is exactly the number of tokens produced, i.e. $\sum_{p \in P} |p^o \cap C| = \sum_{p \in P} |{}^o p \cap C|$. Since trivially we have $\sum_{p \in P} |p^o \cap C| = \sum_{t \in C} |{}^o t|$, and $\sum_{p \in P} |{}^o p \cap C| = \sum_{t \in C} |t^o|$, i.e. the number of places' output arcs is equal to the number of transitions' input arcs, and the number of transitions' output arcs is equal to the number of places' input arcs, then $\sum_{t \in C} |{}^o t| = \sum_{t \in C} |t^o|$.

($\Leftarrow$) If the net is unbounded then there exists a sequence σ such that $M \xrightarrow{\sigma} M'$, where each transition in σ is used exactly once, and $M' = M + \delta_\sigma > M$, then $\delta_\sigma > 0$. Therefore, by similar arguments as those in the previous step of this proof, there exists a set $C_\sigma = \{t \in T | t$ is used in $\sigma\}$, satisfying the liveness conditions, such that $\sum_{t \in C_\sigma} |{}^o t| < \sum_{t \in C_\sigma} |t^o|$. □

The previous theorem suggests a possible algorithm to determine the boundedness of a Petri Net, consisting in verifying if for every set C that satisfies the liveness conditions the referred equality holds; however, the cost of this procedure would be extremely expensive. Instead, verifying the above equality only for the entire set of live transitions T_L can be done in linear time. If it is not satisfied , by lemma 5.1 the net is unbounded. Conversely, note that if $\sum_{t \in T_L} |{}^o t| = \sum_{t \in T_L} |t^o|$ then for every place p, $|p^o \cap T_L| = |{}^o p \cap T_L|$. This holds because, as the net is conflict-free, the number of edges entering one place is not less than the number of edges leaving it, and an extra edge entering a place would imply a place with more output than input edges, a contradiction. Moreover, if $|p^o \cap T_L| = |{}^o p \cap T_L| > 1$ (i.e. p is branched) then ${}^o p = p^o$. We introduce the following

Definition 5.1 *Let $\mathcal{G}_L$ be the graph corresponding to the live subnet of a conflict-free Petri net such that $\sum_{t \in T_L} |{}^o t| = \sum_{t \in T_L} |t^o|$. $\mathcal{G}'_L$ is the graph obtained from $\mathcal{G}_L$ by splitting the nodes corresponding to branched places in the following way: for every transition t_i such that $p \in {}^o t_i$ we introduce a place p_i with ${}^o p_i = p_i^o = \{t_i\}$, p_i is said* new. *Unbranched places of $\mathcal{G}_L$ are left unchanged and called* old.

Lemma 5.2 *If in the graph $\mathcal{G}'_L$ there is an edge that is not part of a cycle, the corresponding net is unbounded.*

Proof. Consider an edge from a place p to a transition t. If p is a new place, it is part of a cycle. If p is old, then, as p is part of P_L, there is at least one edge entering p, but no edge entering p can be part of a cycle. Consider now an edge from a transition t to a place p such that it is not part of a cycle and all the edges entering t are part of cycles (if there is not such an edge, all edges are part of cycles). If t has no input, then clearly the net is

unbounded, and if ${}^o t$ is not empty, then all the places in ${}^o t$ can be filled without p been emptied, so the net is unbounded. □

Lemma 5.3 *Let C be a set of transitions. If $\bigcup_{t\in C} {}^o t \subseteq \bigcup_{t\in C} t^o$ in $\mathcal{G}_L$, then $\bigcup_{t\in C} {}^o t \subseteq \bigcup_{t\in C} t^o$ in $\mathcal{G}'_L$*

Proof. It follows directly from the construction of $\mathcal{G}'_L$ □

Lemma 5.4 *If a conflict-free Petri net is unbounded and $\sum_{t\in T_L} |{}^o t| = \sum_{t\in T_L} |t^o|$, then in the graph $\mathcal{G}'_L$ there exists an edge that is not part of a cycle.*

Proof. First recall that if $\sum_{t\in T_L} |{}^o t| = \sum_{t\in T_L} |t^o|$ then for every place p, $|p^o \cap T_L| = |{}^o p \cap T_L|$. But then in $\mathcal{G}'_L$, for every place p, $|p^o| = |{}^o p| = 1$. This is true because every place in $\mathcal{G}_L$ with $|p^o| = k > 1$ is splitted in k different places in $\mathcal{G}'_L$ with $|p^o| = 1$. Since the net is unbounded, there exists a set C that satisfies the liveness conditions such that $\sum_{t\in C} |{}^o t| < \sum_{t\in C} |t^o|$ and hence $\sum_{p\in P} |p^o \cap C| < \sum_{p\in P} |{}^o p \cap C|$. It follows that there must be a place p in $\mathcal{G}_L$ such that $|p^o \cap C| < |{}^o p \cap C|$. This place p cannot be branched because in that case every transition in ${}^o p$ is also in p^o and hence if such a transition is part of C it contributes equally to ${}^o p$ and p^o. Then p can only be unbranched, so $|p^o \cap C| < |{}^o p \cap C|$ implies that $|p^o \cap C| = 0$ and $|{}^o p \cap C| = 1$, and p is an old place in $\mathcal{G}'_L$. Lets consider the edge entering p in $\mathcal{G}'_L$. If this arc were part of a cycle, this cycle should re-enter C either via a transition or via a place. Both alternatives are impossible, because by Lemma 5.3 C fulfills the liveness conditions (every place in the input of a transition in C is also in the output set of a transition in C) and every place has only one input in $\mathcal{G}'_L$. □

Now we are able to prove the following:

Theorem 5.5 *A conflict free Petri net $\mathcal{P} =< P, T, A, M_0 >$ is bounded iff the live subnet $\mathcal{P}_\mathcal{L}$ of $\mathcal{P}$ fulfills the following properties:*

- *$\sum_{t\in T_L} |{}^o t| = \sum_{t\in T_L} |t^o|$ and*
- *every edge of $\mathcal{G}'_L$ is part of a cycle.*

Proof. The proof is immediate from the previous lemmas. □

The necessary and sufficient conditions of Theorem 5.5 allow us to develop a linear algorithm for determining the boundedness of a net in linear time. This can be done by using very well known algorithms of cycle searching, directly over the graph $\mathcal{G}'_L$ that is constructed in linear time, either by modifying the algorithm in the previous section or in a successive step. The complexity of the final algorithm overcomes that of the algorithm proposed in [11], which consists in similar techniques applied to a different graph, the construction of which requires $O(n^2)$ time.

6 Conclusions

We have exploited the representation of Petri nets as hypergraphs to obtain more efficient algorithms for solving very well known problems. In particular we have proved that the following problems:

- finding the set of firable transitions
- finding the live transitions
- deciding the boundedness of the net

can be solved in time which is linear in the description of the net, while the best previous algorithms require quadratic time. For the first problem we propose an on-line algorithm, which allows to answer queries while the net is modified with no additional computational cost.

We think that this approach for modeling Petri nets may be useful to give a new insight to the subject, especially from the algorithmical point of view, giving raise to new and more efficient algorithms for solving different problems. As open questions we can mention the possibility of obtaining on-line algorithms for the other problems (liveness and boundedness), as well as that of using these or similar data structures for other problems for conflict-free or other kinds of Petri nets.

References

[1] G. Ausiello, A. D'Atri, D. Saccá, Graph algorithms for functional dependency manipulation, *J. ACM*, **30** (1983), 752–766.

[2] G. Ausiello, G.F. Italiano, On-line algorithms for polynomially solvable satisfiability problems, to appear in *Journal of Logic Programming* (1991).

[3] G. Ausiello, G.F. Italiano, and U. Nanni, Dynamic Maintenance of Directed Hypergraphs *Theoretical Computer Science* **72**, 2-3 (1990), 97–117.

[4] C. Berge, *Graphs and Hypergraphs*, North Holland, Amsterdam (1973).

[5] H. Boley, Directed recursive labelnode hypergraphs: A new representation language, *Artificial Intelligence*, **9** (1977), 49–85.

[6] S. Crespi-Reghizzi, and D. Mandrioli, A Decidability Theorem for a Class of Vector Additions Systems, *Information Processing Letters*, **3**, 3 (1975),78–80.

[7] G. Gallo, G. Longo, S. Nguyen, S. Pallottino, Directed Hypergraphs and Applications *Tech. Rep. TR-3/90*, Univ. Pisa, Dip. di Informatica, January 1990.

[8] R. Howell, and L. Rosier, Completeness Results for Reachability, Containment, and Equivalence, with Respect to Conflict-Free Vector Replacement Systems, *Proc. of the 14th International Colloquium on Automata, Languages, and Programming*, Lect. Not. in Computer Sci., **267**, Springer-Verlag (1987), 509–520.

[9] R. Howell, and L. Rosier, Recent Results on the Complexity of Problems Related to Petri Nets, *Advances in Petri Nets 1987*, Lect. Not. in Computer Sci. **266**, Springer-Verlag (1987), 43–72.

[10] R. Howell, and L. Rosier, On Questions of Fairness and Temporal Logic for Conflict-Free Petri Nets, *Advances in Petri Nets 1988*, Lect. Not. in Computer Sci. **340**, Springer-Verlag (1988), 200–226.

[11] R. Howell, L. Rosier, and H.Yen, An $O(n^{1.5})$ Algorithm to Decide Boundedness for Conflict-Free Vector Replacement System, *Information Processing Letters*, **25**, 1 (1987), 27–33.

[12] G.F. Italiano, Amortized efficiency of a path retrieval data structure, *Theoretical Computer Science*, **48**, (1986), 273–281.

[13] M. Jantzen, Complexity of Place / Transitions Nets, *Advances in Petri Nets 1986*, Lect. Not. in Computer Sci. **255**, Springer-Verlag (1986), 411–434.

[14] N. D. Jones, L. H. Landweber, Y. E. Lien, Complexity of some Problem in Petri Nets, *Theoret. Comp. Sci.*, **4**, (1977), 277–299.

[15] R.M. Karp, and R. Miller Parallel Program Schemata, *J. Comput. Syst. Sci.*, **3**, 2 (1969), 147–195.

[16] J.A. La Poutre, J. Van Leeuwen, Maintenance of Transitive Closures and Transitive Reductions of Graphs, *Technical Report RUU-CS-87-25*, University of Utrecht (1987).

[17] L. Landweber, and E. Robertson, Properties of Conflict-Free and Persistent Petri Nets, *J. ACM*, **25**, 3 (1978), 352–364.

[18] J. L. Peterson, Petri Nets, *Computing Surveys*, **9**, 3 (1977), 223–252.

[19] C. A. Petri, Communication with automata, Supplement 1 to Tech.Report RADC-TR-65-377, Vol. 1 Griffiss AFB, New York (1966), original in German; Kommunikation mit Automaten, Univ. of Bonn (1962).

[20] W.Reisig, *Petri Nets, an introduction*, ETACS Monographs Theoret. Comp. Sci., Springer-Verlag (1985).

[21] R.E. Tarjan, Amortized computational complexity, *SIAM J. Alg. Disc. Meth.*, **6** (1985), 306–318.

Appendix

In this appendix we show on-line algorithms to maintain the reachable portion of a conflict-free Petri net while adding new elements to the net, according to the operations described in theorem 3.3.

We will introduce a dummy T-node t_0 in the hypergraph associated to the net, such that for each $p \in M_0$ there is a hyperarc $< \{t_0\}, p >$ in $\mathcal{H}_{\mathcal{P}}$. It will allows us to handle

incremental modifications of the initial marking M_0 in $\mathcal{P}$ as insertions of arcs. Thereafter we say that any P- or T-node y is *reachable* if and only if there exists a hyperpath from t_0 to y in $\mathcal{H}_\mathcal{P}$.

For any T-node t, the P-nodes in ${}^o t$ and t^o are respectively stored in lists; similarly it is done for any P-node p for ${}^o p$ and p^o.

Moreover for any node in $\mathcal{H}_\mathcal{P}$ a counter C is maintained such that:

$C_{t_0} = 0$;

$C_t = |\{p \mid p \in {}^o t \text{ and } p \text{ is not reachable}\}|$;

$C_p = 0$ for any P-node p reachable from t_0, otherwise $C_p = 1$.

The algorithm *make-transition* performs the required updates to the data structures while inserting a transition in the net $\mathcal{P}$ or, more precisely, while inserting a T-node t and a hyperarc in $\mathcal{H}_\mathcal{P}$, given the new transition t and the set ${}^o t$ of P-nodes:

```
algorithm make-transition(X:set of P-nodes, t:T-node);
begin
  C_t ← Σ_{p∈X} C_p;
  °t ← ∅;
  for each p ∈ X do
    begin
      insert p into °t;
      insert t into p°;
    end;
end.
```

We remark that the above algorithm assumes that the new insertion leaves the net conflict-free. The algorithm can be easily modified to perform this test or to insert the required arcs.

The following algorithm correctly updates the data structures while inserting in $\mathcal{P}$ a connection from a transition to a place, that is a hyperarc $< \{t\}, p >$ in $\mathcal{H}_\mathcal{P}$:

```
algorithm insert-arc (t:T-node, p:P-node);
begin
  insert p into t°;
  insert t into °p;
  if C_t = 0
    then closure (t,p);
end.
```

The aim of algorithm *closure* is to update the array C after the insertion of the hyperarc $< \{t\}, p >$. In particular, a $closure(x, y)$ tries to find new reachable nodes:

```
algorithm closure(x,y:node);
begin
```

```
  if C_y ≠ 0
    then begin
           C_y ← C_y − 1;
           if C_y = 0
             then for each z in y^o do
                    closure(y, z);
         end;
end.
```

The algorithm *closure* can be easily modified in order to maintain the reachable portion of the Petri net. In fact, since a node x becomes reachable in $\mathcal{H}_{\mathcal{P}}$ as soon as the value of C_x becames 0, we might update a copy of the reachable portion of the net as soon as a T- or P-node satisfies this condition.

Now we are able to prove theorem 3.3 that can be rephrased in the following way:

The time required to update the data structures described in this appendix, namely the counters C_x, while inserting new transitions and new hyperarc from transitions to places, starting from the empty Petri net using the above algorithms is $O(n_A + n_P + n_T)$ (where the cardinalities are referred to the final net).

Defining new transitions requires constant time for each transition t and for each place $p \in {}^o t$ in the net.

Each pair (x, y) used in a call to the algorithm closure can be used only once (as soon as the node x becomes reachable from t_0). On the other side a call to closure(x, y) requires constant time plus other calls to the algorithm closure (that can be charged to other pairs). This means that the global time spent for each pair (x, y) handled by the algorithm closure is constant.