COMPUTER SCIENCES DEPARTMENT

University of Wisconsin-Madison

COMPLEXITY OF SOME PROBLEMS IN PETRI NETS

by

Neil D. Jones, Lawrence H. Landweber
and Y. Edmund Lien

Computer Sciences Technical Report # 276

September 1976

COMPLEXITY OF SOME PROBLEMS IN PETRI NETS[†]

by

Neil D. Jones

Department of Computer Science
University of Kansas
Lawrence, Kansas 66045

Lawrence H. Landweber*

Computer Sciences Department
University of Wisconsin
1210 W. Dayton

Y. Edmund Lien**

Department of Computer Science
University of Kansas
Lawrence, Kansas 66045

*The work of this author was supported in part by NSF Contract GJ33087.

**The work of this author was supported in part by a Kansas General Research Grant.

[†]A preliminary version of parts of this paper were presented at the Conference on Petri Nets and Related Models, M.I.T., July 1-3, 1975 and in [30].

Complexity of Some Problems in Petri Nets

by

Neil D. Jones
Lawrence H. Landweber
Y. Edmund Lien

Abstract

We consider the complexity of several standard problems for various classes of Petri nets. In particular, the reachability problem, the liveness problem and the k-boundedness problems are analyzed. Some polynomial time and polynomial space complete problems for Petri nets are given. We then show that the problem of deciding whether a Petri net is persistent is reducible to reachability, partially answering a question of Keller. Reachability and boundedness are proved to be undecidable for the Time Petri net introduced by Merlin. Also presented is the concept of controllability, i.e., the capability of a set of transitions to disable a given transition. We show that the controllability problem requires exponential space, even for 1-bounded nets.

1. Introduction

Petri nets have been used to model parallel computation, computer systems and other complex systems [5,24,25,28]. As a modeling tool, Petri nets offer a simple and powerful formalism for the representation of concurrency and the interaction of events in a system. The mathematical properties of Petri nets reflect the properties and patterns of behavior of the systems being modeled. For example, a study of liveness in Petri nets will help us to understand system deadlocks. Boundedness is related to the "storage capacity" required to hold the commodities in a system.

In this paper, we analyze the computational complexity of some problems in Petri nets. Each problem is formulated in the following way: A property statement and a class of Petri nets are given. Our results study the complexity of algorithms which determine whether or not an arbitrary Petri net in the class has the given property. This can involve showing that all algorithms which solve a given problem must have a certain complexity and/or analyzing a given algorithm which solves a problem to obtain an upper bound on the problem's complexity.

We first present the basic terminology and definitions for Petri nets and computational complexity. Many of our results are summarized in Table 1 at the end of the paper.

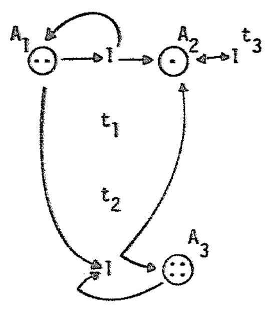
1.1 Petri nets

A Petri net P is a 3-tuple $<P,T,M_0>$ where $P = \{A_1,\dots,A_m\}$ is the finite set of places and $T = \{t_1,\dots,t_n\}$ is the finite set of transitions.

A token distribution, marking or state is a mapping from P Into N, the set of non-negative integers. M_0 is the initial state of P. A state M is represented by $\sum_{i=1}^{m} a_iA_i$ (Σa_iA_i if m is understood.) where a_i is the value $M(A_i)$, the number of tokens in place A_i. A state with no tokens in any place is represented by ϕ. Each transition t_j is denoted by $\Sigma a_iA_i \rightarrow \Sigma b_iA_i$, where a_i and b_i are in $\{0,1\}$. For the transition t_j, the sets of places $I_j = \{A_i | a_i = 1 \text{ in } t_j\}$ and $O_j = \{A_i | b_i = 1 \text{ in } t_j\}$ are referred to as the input places and output places respectively of t_j. Conceptually one may think of a Petri net as a bipartite directed graph (see Figure 1) whose nodes are the places and transitions of the net. Arcs are directed from places to transitions and from transitions to places. Then I_j denotes the set of places having an outgoing arc to t_j and O_j denotes the set of places which have an incoming arc from t_j. The set of input transitions and output transitions of a place can be defined similarly.

The firing of transition t_j results in one token being removed from each place in I_j and one token being added to each place in O_j. Transition t_j can fire at state M if and only if each of its input places contains at least one token at state M. In this case we say that t_j is enabled at M and write $M \overset{t_j}{\Rightarrow} M'$ if the firing of t_j yields state M'. If a transition t_j is not enabled at M, then it is disabled

A firing sequence is a sequence $\sigma = t_{i_1}t_{i_2}\dots t_{i_r} \varepsilon T^*$. A firing sequence is fireable at state M if there are states $M_0,M_1,\dots,M_r$ such that $M = M_0$ and $M_{k-1} \overset{t_{i_k}}{\Longrightarrow} M_k$ for $1 \leq k \leq r$. When a state M' is derived from M via a firing sequence $\sigma \varepsilon T^*$, we write $M \overset{\sigma}{\Rightarrow} M'$. Also,



$\mathcal{P} = \langle P,T,M_0\rangle$

$P = \{A_1,A_2,A_3\}$

$T = \{t_1,t_2,t_3\}$

$M_0 = 2A_1 + 1A_2 + 4A_3$

t_1: $A_1 \to A_1 + A_2$

t_2: $A_1 + A_3 \to A_2 + A_3$

t_3: $\phi \to A_2$

Figure 1. A simple Petri net and its representation as a bipartite directed graph.

$M \overset{*}{\Rightarrow} M'$ if there is a (possibly empty) firing sequence σ such that $M \overset{\sigma}{\Rightarrow} M'$. The reachability set of a Petri net with initial state M_0 is $\{M|M_0 \overset{*}{\Rightarrow} M\}$. The reachability problem for a class of Petri nets is the problem of deciding, given an arbitrary net in the class and an arbitrary state, whether the state is in the reachability set of the net.

A Petri net is k-bounded for an integer $k \geq 1$ if for every state M in the reachability set, $M(A_i) \leq k$ for $1 \leq i \leq m$, i.e., no place in the Petri net will ever receive more than k tokens. A Petri net is safe if it is 1-bounded. A Petri net is bounded if it is k-bounded for some k . Obviously, a Petri net is bounded if and only if it has a finite reachability set.

A transition t_j is live at state M if for any fireable firing sequence σ at M there is a $\tau \epsilon T^*$ such that $\sigma\tau t_j$ is fireable at M . A Petri net is live if every transition of the net is live at the initial state. The liveness problem for a class of Petri nets is the problem of deciding whether an arbitrary net in the class is live.

A Petri net is conservative if there is a positive integer valued function f: $P \to N^+$ such that every transition t_j satisfies

$$\sum_{A_i \epsilon I_j} f(A_i) = \sum_{A_k \epsilon O_j} f(A_k)$$

(where Σ means integer summation). A net is 1-conservative if and only if it is conservative with $f(A_i) = 1$ for all $A_i \epsilon P$. A conservative Petri net has a finite reachability set but the converse is not true. Conservative Petri nets are studied in [18].

A Petri net is a free choice Petri net (FCPN) if for every transition $t_j \epsilon T$ and place $A_i \epsilon I_j$ either t_j is the only output transition of A_i or A_i is the only input place of t_j. A Petri net is persistent if for every state M in the reachability set and any transitions t_i and t_j $(i \neq j)$, if t_i and t_j are enabled at M, then the sequence $t_i t_j$ is fireable at M. A Petri net is ‡conflict-free if each place $A_i \epsilon P$ satisfies either 1) there is at most one arc out of A_i or 2) for all j, $A_i \epsilon I_j$ if and only if $A_i \epsilon O_j$. Note that all conflict free nets are persistent but the converse is not true. Persistent and conflict free Petri nets are studied in [17].

The coverability problem for a Petri net P is defined as follows: Given a state M (not necessarily reachable), is there a reachable state M' of P such that $M' \geq M$ (componentwise).

1.2 Computational complexity

In this paper we present results on the complexity of some Petri net problems. With a standard encoding scheme such as the one in [14], we view each problem as a language recognition problem.

Let $L_1, L_2 \subseteq \Sigma^*$ be languages where Σ is a finite alphabet with at least two members. Let C be a class of languages over Σ. Then

(1) L_1 is recursively reducible to L_2 if the existence of an algorithm for deciding membership in L_2 implies the existence of an algorithm for deciding membership in L_1.

‡Note that our conflict free nets are often called forward conflict free nets.

(2) L_1 is log-space reducible to L_2 if there is a function $f:\Sigma^* \to \Sigma^*$ such that

(a) f is computable by a deterministic Turing machine in log-space, i.e., the machine uses at most $C \log(|x|)$ squares of scratch tape to compute $f(x)$ where x is initially on a read-only input tape, $|x|$ is the length of x and

(b) for all $x \epsilon \Sigma^*$, $x \epsilon L_1$ if and only if $f(x) \epsilon L_2$.

(3) L_1 is C-hard if L_2 is log-space reducible to L_1 for all $L_2 \epsilon C$.

(4) L_1 is C-complete or complete for C if L_1 is C-hard and $L_1 \epsilon C$.

Here we assume the reader is familiar with Turing machines. A good expository reference can be found in [9].

The following useful result can be easily obtained.

Lemma 1.2.1. Let $\tilde{L}$ denote the complement of $L \subseteq \Sigma^*$, i.e., $\tilde{L} = \Sigma^* - L$.

(1) If L_1 is log-space reducible to L_2 and L_2 is log-space reducible to L_3, then L_1 is log-space reducible to L_3.

(2) If L_1 is log-space reducible to L_2, then $\tilde{L}_1$ is log-space reducible to $\tilde{L}_2$.

Proof. See [12].

We shall consider several well known classes of languages. The reader is referred to the presentation in [2] for further details.

Notation

(1) DSPACE (poly) and DSPACE (exp) are used to denote the set of languages recognizable by deterministic Turing machines in polynomial space and exponential space respectively.

(2) NSPACE (log), NTIME (poly), NSPACE (poly) and NSPACE (exp) are used to denote the set or languages recognizable by nondeterministic Turing machines in log space, polynomial time, polynomial space and exponential space respectively.

It is known that DSPACE (poly) = NSPACE (poly), DSPACE (exp) = NSPACE (exp) and NSPACE (log) $\subseteq$ DTIME (poly) $\subseteq$ NTIME (poly) $\subseteq$ DSPACE (poly). It is not known whether any of the above containments is proper.

In Section 2, we consider problems in NTIME (poly). Section 3 deals with problems in DSPACE (poly). In Section 4, we present the first problem in Petri nets known to have at least exponential space complexity. Section 5 shows that persistence is recursively reducible to reachability. Both the decidability of persistence and the decidability of the reachability problem are open. The best known result is an exponential space lower bound for the reachability problem [20]. Section 6 shows that reachability and boundedness are undecidable for the Time Petri net introduced in [21]. Boundedness is known to be decidable for ordinary Petri nets [15].

2. Problems in NTIME (poly)

Commoner introduced the concept of free choice Petri nets (FCPN) and established a necessary and sufficient condition for a FCPN to be live. The result was first reported by Hack [5].

Consider a set D of places of a Petri net. Let

$D^{\bullet} = \{t | t \text{ is an output transition of a place } A \epsilon D\}$

${}^{\bullet}D = \{t | t \text{ is an input transition of a place } A \epsilon D\}$.

If a set D satisfies ${}^{\bullet}D \subseteq D^{\bullet}$, it is called a deadlock Clearly, if a deadlock is blank, i.e., contains no tokens, it will remain blank under transition firings. If a set D satisfies $D^{\bullet} \subseteq {}^{\bullet}D$, it is called a trap. Thus a trap which contains at least one token cannot become blank. Note that the union of two traps is also a trap. Let T_D be the union of all traps contained in the set of places D .

Commoner has shown that a FCPN is live if and only if for each deadlock D with $D^{\bullet} \neq \phi$, $T_D \neq \phi$ and T_D is not blank. To determine if a FCPN is not live, one can look for a deadlock that does not satisfy the above condition. This can be done easily by a nondeterministic Turing machine in polynomial time. Therefore to determine whether a FCPN is not live is a problem in NTIME (poly).

The next theorem asserts that the problem is complete for NTIME (poly).

Theorem 2.1. The liveness problem for FCPN as defined below is complete for NTIME (poly).

Given: a FCPN P

To decide: if P is not live

Proof. It remains to show that an NTIME (poly)-complete problem is log-space reducible to the liveness problem for FCPN. We reduce the satisfiability problem for conjunctive normal form propositional calculus formulas to the liveness problem for FCPN.

Let $x_1,\dots,x_n$ be the propositionalvariables and let $C_1,\dots,C_k$ be the clauses whose conjunction K is to be decided for satisfiability where each C_i is a disjunction of propositional variables x_i and negations of propositional variables, $\bar{x}_i$. We construct a Petri net $P = \langle P,T,M_0\rangle$ with

$$P = \{A_i, x_i, \bar{x}_i \mid x_i \text{ is a propositional variable of } K\}$$
$$\cup \{\langle \bar{x}_i, C_j\rangle \mid \text{ if } x_i \text{ is in } C_j\}$$
$$\cup \{\langle x_i, C_j\rangle \mid \text{ if } \bar{x}_i \text{ is in } C_j\} \cup \{F\}$$

and $M_0(A_i) = 1$ for all A_i , $M_0(p) = 0$ for $p \varepsilon P - \{A_i\}$. The transitions in T are defined as follows (Each line defines one transition t_j , with I_j and O_j denoted by places to the left and right of the arrows respectively.):

(1) For each variable x_i ,

$$\left.\begin{array}{l} A_i \rightarrow x_i \\ A_i \rightarrow \bar{x}_i \end{array}\right\} \qquad (1a)$$

$$\left.\begin{array}{l} x_i \rightarrow \sum\limits_{\bar{x}_i \varepsilon C_j} \langle x_i, C_j\rangle \\ \bar{x}_i \rightarrow \sum\limits_{x_i \varepsilon C_j} \langle \bar{x}_i, C_j\rangle \end{array}\right\} \qquad (1b)$$

(2) for every clause C_j ,

$$\sum_{\bar{x}_i \varepsilon C_j} \langle x_i, C_j\rangle + \sum_{x_i \varepsilon C_j} \langle \bar{x}_i, C_j\rangle \rightarrow F$$

(3) $F \rightarrow \sum\limits_{i=1}^{\ell} A_i$

If $C_1 \wedge C_2 \wedge \dots \wedge C_k$ is satisfiable, then there exists an assignment $f:\{x_1,x_2,\dots,x_n\} \rightarrow \{\underline{\text{true}}, \underline{\text{false}}\}$ such that $C_j = \underline{\text{true}}$ for $1 \leq j \leq k$. The following firing sequence will lead to a state in which no transition can be fired.

Step 1. If $f(x_i) = \underline{\text{true}}$, fire $A_i \rightarrow x_i$. Otherwise, fire $A_i \rightarrow \bar{x}_i$.

Step 2. Fire the appropriate transition in (1b) which are enabled.

After Step 2, no transitions are fireable, because if there is a fireable transition, it must be the one of type (2). However, the assignment f guarantees that at least one of its input places has no token and hence a contradiction is established. Consequently, P is not live.

Assume $C_1 \wedge \ldots \wedge C_k$ is not satisfiable. If each A_i contains at least one token, then obviously every transition in (1) is eventually fireable. Furthermore, by choosing an assignment which makes C_j false and firing the corresponding (1) transitions, the (2) transition for C_j can be enabled. Hence every (2) transition is eventually fireable at M if $M(A_i) \geq 1$ for all A_i. As a consequence (3) is eventually fireable at any such marking. The firing of (3) would then result in a token being placed in each A_i.

Because $C_1 \wedge \ldots \wedge C_k$ is not satisfiable, for any combination of (1) firings beginning at a marking where each A_i contains a token there is a (2) transition that can be enabled (by firing zero or more additional (1) transitions). Then a (2) transition followed by (3) can be fired again leading to a marking for which each A_i contains at least one token. Hence, regardless of earlier firings, a marking M with $M(A_i) \geq 1$ for all A_i can be reached and from this marking every transition is eventually fireable. Because $M_0(A_i) = 1$ for each A_i, P is live.

Note that the Petri net P is a FCPN. Thus we have shown that the satisfiability problem is log-space reducible to the liveness problem for FCPN. Since the liveness problem for FCPN is in NTIME (poly), it is therefore complete for NTIME (poly). □

Example. Let $K = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3)$ be a propositional formula. The reduction in the previous theorem leads to a Petri net P with the following transitions.

$(t_1)\ A_1 \to x_1$

$(t_2)\ A_1 \to \bar{x}_1$

$(t_3)\ A_2 \to x_2$

$(t_4)\ A_2 \to \bar{x}_2$

$(t_5)\ A_3 \to x_3$

$(t_6)\ A_3 \to \bar{x}_3$

$(t_7)\ \bar{x}_1 \to \langle \bar{x}_1, C_1 \rangle + \langle \bar{x}_1, C_2 \rangle$

$(t_8)\ x_2 \to \langle x_2, C_1 \rangle$

$(t_9)\ \bar{x}_2 \to \langle \bar{x}_2, C_3 \rangle$

$(t_{10})\ x_3 \to \langle x_3, C_2 \rangle + \langle x_3, C_3 \rangle$

$(t_{11})\ \bar{x}_3 \to \langle \bar{x}_3, C_1 \rangle$

$(t_{12})\ \langle \bar{x}_1, C_1 \rangle + \langle x_2, C_1 \rangle + \langle \bar{x}_3, C_1 \rangle \to F$

$(t_{13})\ \langle \bar{x}_1, C_2 \rangle + \langle x_3, C_2 \rangle \to F$

$(t_{14})\ \langle \bar{x}_2, C_3 \rangle + \langle x_3, C_3 \rangle \to F$

$(t_{15})\ F \to A_1 + A_2 + A_3$

By the theorem, K is satisfiable if and only if the corresponding FCPN P is not live. In this example, K can be satisfied by assigning x_1 = true, x_2 = true and x_3 = false. The firing sequence $t_1 t_3 t_6 t_8 t_{11}$ leads to state $x_1 + \langle x_2, C_1 \rangle + \langle \bar{x}_3, C_1 \rangle$ in which no transition is live. Note also that any assignment which makes K false corresponds to a firing sequence that places at least one token into F. For example, $t_2 t_4 t_5 t_7 t_9 t_{10} t_{13} t_{14}$ will place two tokens into F.

The reduction in the above proof does not produce a conservative net. Hence for conservative FCPN we have only

Corollary 2.2. The liveness problem for conservative FCPN is in NTIME (poly).

The reachability problem and the liveness problem are closely related. In fact, Hack [7] has shown that for arbitrary Petri nets, the reachability problem and the liveness problem are recursively reducible to each other. Hence the decidability of one problem implies the decidability of the other. Moreover, it is easy to see that the reachability problem for arbitrary nets is log-space reducible to the reachability problem for FCPN. Unfortunately, Hack's proof, recursively reducing reachability to liveness, does not preserve the free choice property. Hence the decidability of liveness for FCPN does not yield the decidability of reachability or liveness for arbitrary Petri nets or the decidability of reachability for FCPN.

We next consider the conservation property of Petri nets. Recall that a Petri net is conservative if there is a positive integer valued function $f: P \to N^+$ such that for every transition t_j

$$\sum_{A_i \in I_j} f(A_i) = \sum_{A_k \in O_j} f(A_k) .$$

Alternatively, if we construct a matrix Γ such that the j^{th} row vector Γ_j corresponds to t_j as follows

$$\Gamma_{ji} = \begin{cases} -1 & \text{if } A_i \in I_j - O_j \\ 1 & \text{if } A_i \in O_j - I_j \\ 0 & \text{otherwise} \end{cases}$$

then positive integer solutions to $\Gamma \cdot x = 0$ correspond to conservation functions where $x_i = f(A_i)$. With this observation in mind, we shall consider the conservation property or Petri nets within the general framework of linear systems.

Let Γ be a matrix of integer entries and let b be an integer vector. We are interested in the following problems:

(A) Given: Γ

To decide: if $\Gamma \cdot x = 0$ has a positive integer solution

(B) Given: Γ, b

To decide: if $\Gamma \cdot x = b$ has a non-negative rational solution

(C) Given: Γ, b

To decide: if $\Gamma \cdot x \geq b$ has a non-negative rational solution

(D) Given: Γ, b

To decide: if $\Gamma \cdot x \geq b$ has a rational solution

We also consider the complements of problems B, C, and D . The complement of a problem is to decide if the linear system in question does not have a solution of the specified type. Denote the complement of problems B, C and D by $\tilde{B}$, $\tilde{C}$ and $\tilde{D}$. For instance $\tilde{B}$ is ($\tilde{B}$)

Given: Γ , b

To decide: if $\Gamma \cdot x = b$ has no nonnegative rational solution.

We observe that the conservation problem for Petri nets is a special case of A . Namely, all nonzero entries in the matrix Γ for a Petri net are in $\{1,-1\}$. Problem D was mentioned in [14] as a problem in NTIME (poly) but not known to be complete for NTIME (poly). In the next result we provide evidence that B is not complete for NTIME (poly) by showing that both D and $\tilde{D}$ are log-space reducible to each other and hence both are in NTIME (poly). In fact we show that any two problems in $\{B,\tilde{B},C,\tilde{C},D,\tilde{D}\}$ are log-space reducible to each other. If any of these problems is complete for NTIME (poly), then the class NTIME (poly) is closed under complement, which seems unlikely. Another problem known to have this property is the primality test problem which is conjectured not to be NTIME (poly) - complete [26].

Problem A is easily shown log-space reducible to B. Therefore the complexity of B serves as an upper bound of the complexity of A.

Lemma 2.3. A is log-space reducible to B.

Proof. Let e be a vector with 1 in each component. $\Gamma\cdot x = 0$ has a positive integer solution if and only if $\Gamma\cdot y = b$ has a nonnegative rational solution for $b = -\Gamma\cdot e$. □

Lemma 2.4. Each of the problems $B,C,D,\tilde{B},\tilde{C}$, and $\tilde{D}$ is log-space reducible to every other.

Proof. By Lemma 1.2.1, it is sufficient to show that

(1) B is log-space reducible to C

(2) C is log-space reducible to D

(3) D is log-space reducible to $\tilde{B}$

(1) $\Gamma\cdot x = b$ has a nonnegative rational solution if and only if $\Gamma\cdot x \geq b$ and $-\Gamma\cdot x \geq -b$, treated as one linear system, has a nonnegative rational solution.

(2) $\Gamma\cdot x \geq b$ has a nonnegative rational solution if and only if $\Gamma\cdot x \geq b$ and $x \geq 0$, treated as one linear system, has a rational solution.

(3) Append the vector $-b$ to the matrix Γ , let the result be Γ' . $\Gamma\cdot x \geq b$ has a rational solution if and only if $\Gamma'\cdot y \geq 0$ has a rational solution with the last component y_k of y being positive. By Theorem 1.2 in [10], $\Gamma'\cdot y \geq 0$ and $y_k > 0$ has a rational solution if and only if $z\cdot\Gamma' = d$ $(d\neq 0)$ has no nonnegative rational solution where $d_k = -1$ and $d_i = 0$ for $1 \leq i \leq k-1$. □

The previous results yield

Theorem 2.5. The following problem is in NTIME (poly).

Given: an arbitrary Petri net or a FCPN P

To decide: if P is conservative

Proof. B is in NTIME (poly) and A is log-space reducible to B . □

For conflict free nets, reachability is known to be decidable [3]. Our next theorem gives an NTIME (poly)-hard lower bound for this problem.

Theorem 2.6. The reachability problem for conflict free Petri nets is NTIME (poly)-hard.

Proof. We reduce the satisfiability probem for conjunctive normal form propositional calculus formulas to the reachability problem for conflict free nets. Let $K = C_1 \wedge \ldots \wedge C_n$ be a CNF formula where each C_i is a disjunction of (some of) the variables $x_1,\ldots,x_r$ and their negations $\bar{x}_1,\ldots,\bar{x}_r$. We construct a Petri net $P = <P,T,M_0>$ with $P = \{x_1,\ldots,x_r,C_1,\ldots,C_n\}$, i.e., one place for each conjunct and one place for each variable of K. The transitions in T are defined as follows:

(1) for each variable x_i, $1 \leq i \leq r$

$$\phi \to \sum_{x_i \varepsilon C_j} C_j + x_i$$

(2) for each negation of a variable $\bar{x}_i$, $1 \leq i \leq r$

$$\phi \to \sum_{\bar{x}_i \varepsilon C_j} C_j + x_i$$

(3) for each conjunct C_j, $1 \leq j \leq n$, $C_j \to \phi$

Let each place initially have zero tokens. We claim that a state for which each place has 1 token is reachable iff K is satisfiable.

Each transition in (1) or (2) puts a token in a place corresponding to a variable x_i. No transition removes these tokens so for a state

with 1 token in x_i to be reached, there must be exactly one firing of the transition in (1) or the transition in (2) for x_i but not both. If (1), then x_i is <u>true</u> and if (2), then x_i is <u>false</u> for this truth valuation. If $\phi \to \sum_{x_i \varepsilon C_j} C_j + x_i$ is fired, then places corresponding to conjuncts made <u>true</u> by x_i = <u>true</u> receive a token. Similarly for x_i = <u>false</u> and transitions in (2). Hence all places can be made greater than 0 iff there is a firing sequence corresponding to a truth valuation for which the formula is <u>true</u>. Transitions in (3) merely reduce the number of tokens in places corresponding to conjuncts to 1 (in case a valuation makes more than one disjunct of a conjunct <u>true</u>). □

3. Polynomial space problems

The membership problem for context sensitive languages is complete for DSPACE (poly) [22]. The problem is usually formulated as follows:

Given: a nondeterministic linear bounded automaton M and a sentence x

To decide: if M accepts x

In order to prove that a problem L is complete for DSPACE (poly), we will show that L is in DSPACE (poly) and that the membership problem for context sensitive languages is reducible to L.

Theorem 3.1. The following problem is complete for DSPACE (poly).

Given: a 1-conservative Petri net $P = \langle P,T,M\rangle$, and a state M'

To decide: if $M \overset{*}{\Rightarrow} M'$

Proof. A nondeterministic Turing machine which maintains one counter for each place, generates a random firing sequence and maintains the current marking on its tape can solve the reachability problem for P. Since the total number of tokens in the places of a 1-conservative Petri net does not change as a result of firing a transition, the required Turing machine can operate in linear space. Consequently, the problem is in DSPACE (poly).

To show it is DSPACE (poly)-hard, suppose we are given a nondeterministic linear bounded automaton $M = \langle K,\Sigma,\Gamma,\delta,q_1,F,\$\rangle$ where K is the set of states, Σ is the set of input symbols, $\Gamma \supseteq \Sigma \cup \{\$\}$ is

the set of tape symbols, $\delta \subseteq K \times \Gamma \times \{C,R,L\} \times K \times \Gamma$ is the state transition relation, $q_1 \in K$ is the initial state, $F \subseteq K$ is the set of final states and $\$ \in \Gamma - \Sigma$ is the boundary symbol of the tape. Let $\Gamma = \{a_1,a_2,\ldots,a_p\}$, $\$ = a_1$ and $K = \{q_1,q_2,\ldots,q_m\}$. Let $x = \$x_1x_2\ldots x_n\$$ be a sentence in Σ^*. We shall construct a 1-conservative Petri net $P = \langle P,T,M\rangle$ as follows:

(a) $P = \{A_{i,j} | 0 \le i \le n+1,\ 1 \le j \le p\}$ $\{Q_{i,j} | 0 \le i \le n+1,\ 1 \le j \le m\}$ $\cup \{C,D\}$ ($A_{i,j}$ will have one token iff the symbol in location i of M's tape is a_j. Similarly $Q_{i,j}$ will have one token iff M is in state q_j scanning location i.)

(b) $M = \sum\limits_{\substack{x_i = a_j \\ 1 \le i \le n}} A_{i,j} + A_{0,1} + A_{n+1,1} + Q_{1,1}$

Corresponding to the starting configuration of M.

(c) Transitions in T are defined according to δ.

(1) If $(q_s,a_t,C,q_r,a_\ell) \in \delta$, [no motion]

then T includes

$Q_{i,s} + A_{i,t} \to Q_{i,r} + A_{i,\ell}$ for all $0 \le i \le n+1$.

(2) If $(q_s,a_t,R,q_r,a_\ell) \in \delta$, [move right]

then T includes

$Q_{i,s} + A_{i,t} \to Q_{i+1,r} + A_{i,\ell}$ for all $0 \le i \le n$.

(3) If $(q_s,a_t,L,q_r,a_\ell) \in \delta$, [move left]

then includes

$Q_{i,s} + A_{i,t} \to Q_{i-1,r} + A_{i,\ell}$ for all $1 \le i \le n+1$.

(4) If q_s is a final state, then T includes

$$Q_{i,s} \to C \qquad \text{for all } 0 \le i \le n+1 .$$

(5) T also includes

$$C + A_{i,j} \to C + D \qquad \text{for all } 1 \le i \le n,\ 1 \le j \le p .$$

Let $M' = C + nD + A_{0,1} + A_{n+1,1}$ (recall that n is a constant). If M can reach a final state q_s and hence accepts the input string x, the Petri net P can simulate the behavior of M by firing transitions in (1), (2), and (3). Finally P fires $Q_{i,s} \to C$ for some i and transitions in (5) to reach the state M'. On the other hand if P can reach state M', the transition t in (4) must have been fired to produce a token in C. Before that, transitions in (5) cannot be fired. Therefore, P must have fired a sequence of transitions in (1), (2), and (3) to make t fireable. The firing sequence corresponds to a sequence of moves of M to accept x. We conclude that M accepts x iff P can reach M'.

It is easy to see that P is 1-conservative and the reduction can be performed in logarithmic space. □[‡]

The same proof can be used to show:

Corollary 3.2. The coverability problem for 1-conservative Petri nets is DSPACE (poly)-complete.

[‡]Note that C. A. Petri's thesis [25] uses a similar Turing machine simulation for different purposes.

Corollary 3.3. The reachability problem for bounded Petri nets is DSPACE (poly)-hard. The reachability problem for k-bounded Petri nets as defined below is DSPACE (poly)-complete.

Given: a Petri net P, a state M and a constant k such that P is k-bounded (assume k is given in binary notation)

To decide: if P can reach M

Corollary 3.4. The k-boundedness problem as defined below is DSPACE (poly)-complete.

Given: a Petri net P and a constant k

To decide: if P is k-bounded

Proof. The construction in the theorem can be modified to produce a Petri net which becomes unbounded if and only if M accepts x. □

The next theorem gives a DSPACE (poly)-hard lower bound for deciding persistence of arbitrary Petri nets. It is not known whether this problem is decidable.

Theorem 3.5. The problem of deciding whether an aribtrary Petri net is persistent is DSPACE (poly)-hard.

Proof. The membership problem for deterministic linear bounded automata is also known to be DSPACE (poly)-complete. Delete the transitions of (5) in the proof of Theorem 3.1 and add transitions

$Q_{is} \rightarrow E$

for all $0 \leq i \leq n+1$ and each final state q_s, where E is a new place. Also assume that the original linear bounded automaton M is deterministic. Then the net obtained is persistent if and only if M does not reach a final state. If a final state q_s is reached while scanning the i-th tape symbol, then both

$Q_{is} \rightarrow E$

and

$Q_{is} \rightarrow C$

are enabled but only one can fire so in the case the net is not persistent. Because M is deterministic, all other transitions preserve persistence. □

Another interesting observation is that for a Petri net P and a state M, we can construct a FCPN P' by modifying transitions such that P can reach M iff P' can. P' is called the "released form" of P in [6].

Theorem 3.6. The reachability problem for a 1-conservative FCPN is DSPACE (poly)-complete.

Proof. For a Petri net $P = \langle P,T,M \rangle$, construct a FCPN P' as follows:

(a) P' has places $\{\langle A_i,t_j \rangle | \text{ for all } t_j \epsilon T,\ A_i \epsilon I_j\} \cup \{A_i | A_i \epsilon P\}$

(b) P' has the following transitions:

(1) For each $t_j \epsilon T$

$$\underset{\text{in } P}{\sum_{A_i \epsilon I_j}} \langle A_i,t_j \rangle \rightarrow \underset{\text{in } P}{\sum_{A_k \epsilon O_j}} A_k$$

(2) If A_i is in I_j, then transitions of P' include

$A_i \rightarrow \langle A_i,t_j \rangle$.

(c) P and P' have the same initial state M.

It can be seen that, given a state M of P, P can reach M iff P' can reach a state M' whose projection onto the places of P is M and where $M'(\langle A_i,t_j \rangle) = 0$ for all i,j. Furthermore, P' is 1-conservative iff P is. Therefore the reachability problem for an arbitrary 1-conservative FCPN is DSPACE (poly)-complete. □

Corollary 3.7. The coverability problem for 1-conservative FCPN is DSPACE (poly)-complete.

Theorem 3.8. The k-boundedness problem for (free choice) conservative Petri nets is DSPACE (poly)-complete.

Proof. First note that k-boundedness can be decided in polynomial space for arbitrary nets and hence this is also true of (free choice) conservative nets. The net obtained in the proof of Theorem 3.1 can be modified so that it is not k-bounded, but is still conservative, in case M accepts its input x. To do this, eliminate the transitions of (5) in the proof of Theorem 3.1. Add transitions

$C \rightarrow D_1 + \ldots + D_{k+1}$

$D_i \rightarrow E \quad (1 < i \leq k+1)$

where $E, D_1, \ldots, D_{k+1}$ are new places.

The function which shows that the net thus obtained is conservative is $f(A) = k + 1$ for all places $A \notin \{D_i\} \cup \{E\}$ and $f(E) = f(D_i) = 1$ for $1 \leq i \leq k + 1$.

The construction of Theorem 3.6 can then be used to obtain a FCPN having the required properties. Notice that the modified net will still be bounded since it is conservative (though not 1-conservative). □

Remark. By the construction of Theorem 3.6, the reachability problem of a general Petri net can be recursively reduced to the reachability problem for a FCPN. The former was proved to be recursively equivalent to the liveness problem for general Petri nets [7]. The construction in Theorem 2 of [7] can be adjusted to produce a FCPN. Therefore, the zero reachability problem for a FCPN is also recursively equivalent to any of the three problems mentioned above. The solvability of these problems remains open.

Theorem 3.9. The liveness problem for 1-conservative Petri nets is in DSPACE (poly).

Proof. Note that the total number of tokens in a 1-conservative Petri net is not changed by the firing of a transition. In order to determine if a 1-conservative Petri net $P = \langle P,T,M_0 \rangle$ is not live, a nondeterministic Turing machine T guesses a transition t , a state M_1 and a sequence σ such that $M_0 \overset{\sigma}{=>} M_1$ and checks "systematically" for any

state M_2 reachable from M_1 whether t is enabled in M_2 . The machine T stops and answers that $\cdot T$ is not live if in every state M_2 , reachable from M_1 , t is not enabled. The standard technique in [27] can be used to check all states reachable from M_1 . In order to use this technique, T uses n registers of size n where n is the space required to record any state in the reachability set. Since P is 1-conservative, n is a polynomial function of the size of P . Hence the liveness problem for 1-conservative Petri nets can be decided in nondeterministic space $O(n^2)$, or deterministic space $O(n^4)$. □

4. Controllability requires exponential space

Petri nets are typically used to model constructs, such as operating systems, in which events occur asynchronously in sequences which are unpredictable but which may affect the state of the entire system. An essential concept for understanding such systems in practice is that of control, i.e., the ability of actions by one part of the system to determine events in another regardless of other concurrent system activities. In this section we formalize this concept in terms of Petri nets and show that determining whether one part controls another is inordinately difficult, requiring at least exponential space. Furthermore, this bound applies even when the problem is restricted to 1-conservative Petri nets, in which tokens are never created or destroyed but merely move from one place to another.

Let $P = <P,T,M>$ be a Petri net. Let T_0 be a subset of T and $\bar{t}$ a transition in $T - T_0$. An erasing homomorphism h can be defined for sequences in T^* such that $h(t) = t$ if $t \in T_0$ and $h(t) = \varepsilon$ if $t \notin T_0$. We say that T_0 <u>controls $\bar{t}$ by a firing sequence</u> x in T_0^* if for every firing sequence σ at state M, $h(\sigma) = x$ implies $\bar{t}$ is not fireable at M. Namely, T_0 can control $\bar{t}$ in the sense that once the sequence x has been fired, even when the transitions of x are interleaved with transitions in $T - T_0$, $\bar{t}$ cannot be made fireable until transitions in T_0 fire again. Further, we say <u>T_0 can control $\bar{t}$</u> if T_0 can control $\bar{t}$ by at least one sequence x.

The controllability problem is defined as follows.

Given: a 1-conservative Petri net $<P,T,M>$, $T_0 \subseteq T$, $\bar{t} \in T - T_0$

To decide: if T_0 can control $\bar{t}$

Theorem 4.1. The controllability problem for 1-conservative FCPN requires at least exponential space. Moreover, it can be solved within exponential space.

Proof. We shall first display a method of constructing a 1-conservative Petri net whose behavior simulates the generation of sentences in a regular subset of $\{0,1\}^*$ which is given by a regular expression with the squaring operation [22]. The square of a set $S \subseteq \{0,1\}^*$ is defined by $S^2 = \{x \cdot y \mid x,y \in S\}$. Exponential space is required to determine whether the complement of a set denoted by a regular expression with squaring is empty (Corollary 2-1 of [22]).

Let R be a regular expression with squaring. We also use R to represent the subset of $\{0,1\}^*$ denoted by the regular expression R. The Petri net P_R, corresponding to R, has special places ZERO, ONE and TAPE. There are also special places IN_{R_1} and OUT_{R_1} for each subexpression of R including R itself. The transitions of P_R are $T_R \cup \{0,1,2\}$ where transitions 0,1 and 2 are defined as follows:

(0) TAPE $\rightarrow$ ZERO

(1) TAPE $\rightarrow$ ONE

(2) $OUT_R \rightarrow IN_R$

The transitions in T_R are described recursively.

(1) If $R = 0$, then T_R includes

$ZERO + IN_R \rightarrow OUT_R + TAPE$

(2) If $R = 1$, then T_R includes

$ONE + IN_R \rightarrow OUT_R + TAPE$

(3) If $R = R_1 \cdot R_2$, then T_R includes

$IN_R \rightarrow IN_{R_1}$

$OUT_{R_1} \rightarrow IN_{R_2}$

$OUT_{R_2} \rightarrow OUT_R$

and all transitions in T_{R_1} and T_{R_2} .

(4) If $R = R_1 \cup R_2$, then T_R includes

$IN_R \rightarrow IN_{R_1}$

$IN_R \rightarrow IN_{R_2}$

$OUT_{R_1} \rightarrow OUT_R$

$OUT_{R_2} \rightarrow OUT_R$

and all transitions in T_{R_1} and T_{R_2}

(5) If $R = R_1^*$, then T_R includes

$IN_R \rightarrow IN_{R_1}$

$IN_{R_1} \rightarrow OUT_R$

$OUT_{R_1} \rightarrow IN_{R_1}$

and all transitions in T_{R_1} .

(6) If $R = R_1^2$, then T_R includes

$IN_R \rightarrow IN_{R_1}$

$OUT_{R_1} + A_R \rightarrow IN_{R_1} + B_R$

$OUT_{R_1} + B_R \rightarrow OUT_R + A_R$

and all transitions in T_{R_1} .

We call A_R and B_R of (6), <u>type A</u> and <u>type B</u> places respectively. To complete the definition of P_R , let P_R be the set of all places occurring in the transitions above, and let the initial token distribution M_R assign one token to TAPE, I_R and all type A places.

Let T_0 be the two transitions 0 and 1 . We define $h(0) = 0$, $h(1) = 1$ and $h(t) = \varepsilon$ for all transitions not in T_0 . It is easily verified that firings of P_R simulate the generation of strings in R in the following sense. A stirng $x \in \{0,1\}^*$ is in R if and only if $x = h(\sigma)$ for some firing sequence σ such that $M_R \overset{\sigma}{\Rightarrow} M'$ for a state M' with $M'(OUT_R) \neq 0$ (i.e., which moves a token from IN_R to OUT_R).

Now let $\bar{t}$ be transition 2 and suppose the transitions in T_0 produce a sentence $x \in \{0,1\}^*$. If x is in R , then there exists a firing sequence of transitions which moves the token from IN_R to OUT_R and makes the transition $\bar{t}$ fireable. If x is not in R , then no matter how the transitions in (1) through (6) are fired, no token can be added to OUT_R and hence $\bar{t}$ is controlled by T_0 .

Further, it can be seen that if T_0 can control $\bar{t}$ by the sentence

x, then x must not be in R. We conclude that T_0 can control $\mathcal{E}$ iff some sentence in $\{0,1\}^*$ is not in R, i.e., the complement of R is not empty.

Then by [22], the controllability problems requires at least exponential space.

By exhaustive enumeration of firing sequences, controllability for 1-conservative Petri nets can be determined within exponential space, so the complexity bound is tight. Furthermore, by using the technique of Theorem 3.6 we can construct a FCPN P' to simulate P so the controllability problem for FCPN also requires exponential space. □

The controllability problem was the first Petri net problem known to require exponential space. Lipton [20] independently showed that the reachability problem for general Petri nets requires exponential space. Since it may be easily shown that reachability is log-space reducible to controllability, Lipton's result implies that deciding controllability requires exponential space for general Petri nets. However, our result establishes the exponential space lower bound for a much more limited class, namely the 1-conservative free choice nets, in which the total number of tokens never changes from the number present in the initial state. Also, controllability for 1-conservative free choice nets can be decided within exponential space. It is reasonable to expect that the controllability and reachability problems for general nets if decidable, will be more difficult to solve, i.e., they probably will not be solvable within exponential space.

5. Persistence

The results of Landweber and Robertson [17] show that persistence is an important property of Petri nets. For example, reachability sets of persistent nets are semi-linear. Moreover, every net is equivalent to one in which non-persistence occurs at no more than two transitions. Because of these facts and the importance of semi-linearity in earlier Petri net work, we believe that a thorough understanding of the role of persistence will be helpful in solving the difficult open Petri net problems.

In this section we show that persistence is recursively reducibile to reachability, i.e., if there is a decision procedure for the reachability problem, then there is a decision procedure which determines whether an arbitrary net is persistent. The decidability of both problems is open. Lipton [20] has given an exponential space lower bound for reachability and in Section 3 we showed that persistence is DSPACE (poly)-hard.

Theorem 5.1. Persistence is recursively reducible to reachability.

Proof. A Petri net P is not persistent if and only if there is a reachable state M and two transitions t_i and t_j which satisfy

$$(\ast)\qquad \begin{cases} M(D) \geq 1 & \text{for all places } D \in (I_i \cup I_j) \\ M(A) = 1 & \text{for some } A \in (I_i \cap I_j) - (O_i \cap O_j) \end{cases}$$

For each pair of transitions t_j, t_i and each $A \varepsilon (I_i \cap I_j) - (O_i \cap O_j)$ construct a Petri net $P_{ij}(A)$ as follows:

1. $P_{ij}(A)$ includes all places, arcs and transitions of P.
2. Add a place B which initially has one token.
3. Add a transition t having no output places and input places $I_i \cup I_j \cup \{B\}$.
4. For each place A', other than A and B, add a transition $t(A')$ with $O_{t(A')} = \emptyset$, $I_{t(A')} = A'$.

Notice that the transition t can fire at most once. The transitions of 4 can be used to remove tokens from places other than A. The net $P_{ij}(A)$ reaches the zero marking iff condition (*) is true for t_i, t_j and A. Hence P is not persistent iff some $P_{ij}(A)$ reaches the zero marking; so if reachability were decidable, persistence would also be decidable. $\square$

Theorem 5.1 partially answers a question raised by Keller [16]. A similar proof appears in [6] where reachability is also shown to be reducible to the persistence of a given transition or pair of transitions. We conjecture that reachability is also reducible to persistence.

6. The Time Petri net

In [21], Merlin introduced a variant of the Petri net model having a weak timing mechanism. In this section, we show that some of the problems considered in previous sections are undecidable for this model. This is unfortuante because the Time Petri net possesses some interesting properties, notably with respect to recoverability as studied by Merlin. Our result is perhaps indicative of why significant results regarding the mathematical properties of Petri nets have been so difficult to obtain. In particular, the computational power of Petri nets seems to lie in an unexplored region between that of finite automata and Turing machines. Moveover, any significant strengthening of the model seems to lead to equivalence with Turing machines. Similar observations and related results appear in [1,4].

A Time Petri net (TPN) is a Petri net plus a timing mechanism. Associated with each transition $t \varepsilon T$ is a pair of numbers (a_1, a_2) $(a_1, a_2 \varepsilon \{\text{real numbers}\} \cup \{\infty\})$. Assume a system clock which counts off time beginning with zero. Further assume that t becomes enabled at time a. Then t may not be fired until time $a + a_1$. Moreover, t must be fired by $a + a_2$ (unless it is disabled before then). Assume that the firing of a transition takes 0 time and further assume that the TPN blocks or is undefined on all computation paths which disobey the above requirements.

We show that the TPN can simulate deterministic input-free 2-counter machines. Since halting is undecidable for such machines, this yields the undecidability of various TPN properties. A related result and construction appears in [4].

An <u>input-free</u>, <u>2-counter machine</u> is a 6-tuple

$$C = \langle Q, q_0, q_F, I, C_1, C_2 \rangle$$

where Q is a finite set of states; $q_0 \varepsilon Q$ is the initial state; $q_F \varepsilon Q$ is the final or halting state; I is a finite set of instructions and C_1 and C_2 are counters, each of which is capable of storing a non-negative integer. The counters are intially set to 0. Instructions are of the form:

a. $(q, D_i, \bar{q})$ $i = 1,2$; $q, \bar{q} \varepsilon Q$
b. $(q, I_i, \bar{q})$ $i = 1,2$; $q, \bar{q} \varepsilon Q$
c. (q, T_i, r, s) $i = 1,2$; $q, r, s \varepsilon Q$.

The instructions are interpreted as follows

a. $(q, D_i, \bar{q})$: in state q, decrement C_i by 1 and go to $\bar{q}$
b. $(q, I_i, \bar{q})$: in state q, increment C_i by 1 and go to $\bar{q}$
c. (q, T_i, r, s): in state q, test C_i; go to r if C_i is empty and go to s otherwise.

Assume the machines are deterministic so that there is at most one instruction for each $q \varepsilon Q$. A 2-counter machine halts if it reaches state q_F . (We assume no instruction begins with q_F .) Computations which end by branching to a non-existent state or by attempting to decrement

an empty counter are undefined. The following is well known.

Lemma 6.1. The halting problem for deterministic 2-counter machines (with counters initially 0) is undecidable.

Proof. 2-counter machines can simulate Turing machines [23]. □

Our next theorem is proved by showing that the TPN can simulate arbitrary deterministic 2-counter machines.

Theorem 6.2. The following properties of the TPN are undecidable.

1. reachability
2. boundedness.

Proof. Let $C = \langle Q, q_0, q_F, I, C_1, C_2 \rangle$ be a deterministic 2-counter machine. To simplify the notation assume instructions of type a and b satisfy $q \neq \bar{q}$ and those of type c satisfy $q \not\varepsilon \{r,s\}$. Any 2-counter machine can be modified to satisfy this property without affecting whether or not it halts when started with its counters empty.

The TPN which simulates C has one place A_q for each state $q \varepsilon Q$. The place for q will have one token when C is in state q and zero tokens otherwise. Initially A_{q_0} has one token and the other state places have zero tokens. There is one place for each counter, A^1 for C_1 and A^2 for C_2 . Initially A^1 and A^2 have zero tokens. There is also one place A'_q for each instruction of the form (q, T_i, r, s) .

Initially this place has no tokens.

Instructions of C are simulated by transitions in T and associated times.

$\underline{I}$	$\underline{T}$	
$(q,D_i,\bar{q})$	$A_q + A^i \to A_{\bar{q}}$	
	$a_1 = 0,\ a_2 = \infty$	
$(q,I_i,\bar{q})$	$A_q \to A_{\bar{q}} + A^i$	
	$a_1 = 0,\ a_2 = \infty$	
(q,T_i,r,s)	$A_q + A^i \to A'_q$	$a_1 = 0,\ a_2 = 1$
	$A'_q \to A_s + A^i$	$a_1 = 0,\ a_2 = \infty$
	$A_q \to A_r$	$a_1 = 2,\ a_2 = 3$

The transitions corresponding to instructions which increment or decrement counters can be fired whenever they are enabled. If an instruction tests a counter, there are three transitions associated with it. The first two correspond to the counter not empty case and the third to the counter empty case. If the counter is not empty, the first transition will be fired, disabling the third. This occurs because of the times associated with the two transitions. The second transition resets the counter C_i to the correct value. If the first transition is not enabled, then eventually the third transition will be fired. It should be clear that the TPN P, as defined above, simulates the 2-counter machine C. Moreover, P reaches a marking with a single token in place Aq_F iff C halts. Because of our assumption that no instruction begins

with q_F, it follows that no transition is fired after Aq_F receives a token.

1. Reachability: A marking with one token in Aq_F is reached iff C halts. Add transitions which empty all places if a state with a token in Aq_F is reached. Add transitions:

$$A_{q_F} + A' \to A_{q_F}$$

$$A_{q_F} + A^2 \to A_{q_F}$$

$$A_{q_F} \to \emptyset$$

all with associated times 0 and ∞. Then the zero marking is reached in the modified TPN iff q_F is reached in C iff C halts.

2. Boundedness: Add an additional place d which initially has no tokens. Modify each transition of P to add a token to d. Then the number of tokens d receives is bounded iff the length of fireable firing sequences in the modified P is bounded iff C halts. □

PETRI NET CLASSES / PROBLEMS

		reachability	liveness	coverability	k-boundedness	conservation
State Machine Graph		NSPACE (log) complete	NSPACE (log) complete	NSPACE (log) complete	(not k-bounded) NSPACE (log) complete	**trivial**
Marked Graph		(not reachable) NSPACE (log) hard	(not live) NSPACE (log) complete	(not coverable) NSPACE (log) hard	(not k-bounded) NSPACE (log) hard	**NSPACE (log) complete**
Free Choice Petri Net	1-conservative	DSPACE (poly) complete	(not live) NTIME (poly)	DSPACE (poly) complete	DSPACE (poly) complete	**trivial**
	any	DSPACE (exp) hard	(not live) NTIME (poly) complete	DSPACE (exp) hard	DSPACE (poly) complete	**NTIME (poly)**
Petri Net	1-conservative	DSPACE (poly) complete	DSPACE (poly)	DSPACE (poly) complete	DSPACE (poly) complete	**trivial**
	any	DSPACE (exp) hard	DSPACE (exp) hard	DSPACE (exp) hard	DSPACE (poly) complete	**NTIME (poly)**

Table 1. Summary of some complexity results for Petri nets. The DSPACE (exp)-hard results are due to Lipton [20].

7. Conclusion

In Table 1, we summarize some known results about the complexity of problems in Petri nets. For each class of Petri nets listed in the first column, five problems are considered: reachability, liveness, coverability, k-boundedness, and conservation. Further explanation of a problem is given in parentheses in the entry. For instance, (not live) means that the property to decide is that the given Petri net is not live.

The results on state machine graphs and marked graphs follow directly from earlier work on NSPACE (log)-complete and NSPACE (log)-hard problems [12, 13].

The most important open problems on Petri nets involve the finding of decision procedures (if such exist) for reachability, liveness and persistence. Before these problems can be solved, we must develop a greater understanding of the characteristics of Petri nets. We believe that one way to achieve this is to study the complexity of deciding various Petri net properties. In this paper we have investigated such questions for a variety of problems and types of Petri nets. Our results indicate that many problems concerning Petri net behavior are intrinsically very hard to solve. Consequently any algorithms which analyze the types of Petri net behavior discussed here will in the worst cases require unacceptable amounts of computation time or space.

References

[1] T. Agerwala, A complete model for representing the coordination of asynchronous processes, Computer Research Report 32, Johns Hopkins University (July 1974).

[2] R. V. Book, Comparing complexity classes, JCSS 9 (1974) 213-229.

[3] S. Crespi-Reghizzi and D. Mandrioli, A decidability theorem for a class of vector addition systems, Inf. Proc. Letters 3 (1975) 78-80.

[4] M. Hack, Petri net languages, Comp. Structures Group Memo 124 M.I.T. Project MAC (June 1975).

[5] M. Hack, Analysis of production schemata by Petri nets, MAC TR-94, Project MAC, M.I.T., Cambridge, Massachusetts, (1972).

[6] M. Hack, Extended state machine allocatable (ESMA) nets, an extension of free choice Petri net results, Comp. Structures Group Memo 78, Project MAC, M.I.T. (1973).

[7] M. Hack, The recursive equivalence of the reachability problem and the liveness problem for Petri nets and vector addition systems, 15th Annual Symposium on Switching and Automata Theory (IEEE 1974) 156-164.

[8] M. Hack, Decidability questions for Petri nets, Ph.D. Thesis, M.I.T. (December 1975).

[9] J. Hopcroft and J. Ullman, Formal Languages and their Relation to Automata (Addison-Wesley, Reading, Mass., 1969).

[10] T. C. Hu, Integer Programming and Network Flows (Addison-Wesley, Reading, Mass. (1969).

[11] N. D. Jones and Y. E. Lien, Complexity of some problems in Petri nets, Conf. on Petri Nets and Related Models, M.I.T. (July 1975).

[12] N. D. Jones, Space-bounded reducibility among combinatorial problems, JCSS 11 (1975) 68-85.

[13] N. D. Jones, Y. E. Lien and W. T. Laaser, New problems complete for nondeterministic log space, to appear in Mathematical Systems Theory.

[14] R. M. Karp, Reducibility among combinatorial problems, in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds. (Plenum Press, New York, 1972) 85-103.

[15] R. M. Karp and R. E. Miller, Parallel program schemata, JCSS 3 (1969) 147-195.

[16] R. M. Keller, Vector replacement systems: a formalism for modelling asynchronous systems, TR 117, Princeton University Computer Science Laboratory (December 1972).

[17] L. H. Landweber, and E. L. Robertson, Properties of conflict free and persistent Petri nets, University of Wisconsin Computer Science Department Technical Report 264 (December 1975).

[18] Y. E. Lien, A note on transition systems, to appear in J. of Information Sciences.

[19] Y. E. Lien, Termination properties of generalized Petri nets, SIAM J. Computing, 5 (1976) 251-265.

[20] R. Lipton, The reachability problem and the boundedness problem for Petri nets are exponential-space hard, Conference on Petri Nets and Related Methods, M.I.T. (July, 1975), Also, Yale Research Report # 62 (1976).

[21] P. M. Merlin, A study of the recoverability of computing systems, Ph.D. Thesis, U.C.-Irvine (1975).

[22] A. R. Meyer and L. Stockmeyer, The equivalence problems for regular expressions with squaring requires exponential space, 13th Annual Symposium on Switching and Automata Theory (IEEE 1972) 125-129.

[23] M. Minsky, Recursive unsolvability of Post's problem, Annals of Math 74 (1961) 437-454.

[24] J. D. Noe, A Petri net model of the CDC 6400, Proc. of the ACM/SIGOPS Workshop on Systems Performance Evaluation (ACM 1971) 362-378.

[25] 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).

[26] V. R. Pratt, Every prime has a succinct certificate, SIAM Journal on Computing 4, (1975) 214-220.

[27] W. J. Savitch, Relations between nondeterministic and deterministic tape complexities, JCSS 4 (1970) 177-192.

[28] R. M. Shapiro and H. Saint, A new approach to optimization of sequencing decisions, Annual Review of Automatic Programming (Pergamon Press, Oxford, England, 1970) Vol. 6 Part 5.

[29] L. J. Stockmeyer and A. R. Meyer, Word problems requiring exponential time: Preliminary Report, Fifth ACM Symp. Thy. Computing (ACM 1973) 1-9.

[30] L. H. Landweber, Properties of vector addition systems, Univ. of Wisc. Comp. Sci. Dept. Tech. Report 258 (June 1975).