

PETRI NET LANGUAGES AND THEIR APPLICATIONS

by

M. Yoeli and A. Ginzburg

Department of Computer Science

Technion, Haifa, Israel

Research Report CS-78-45

November 1978

This work was done whilst the authors were visiting the University of Waterloo. The research work was supported by Natural Sciences and Engineering Research Council Canada grant A-1617.

PETRI NET LANGUAGES AND THEIR APPLICATIONS

by

M. Yoeli and A. Ginzburg

Abstract

This report provides an easy introduction to Petri nets and their languages and demonstrates the applicability of Petri net languages to the study of systems involving concurrency.

PETRI NET LANGUAGES AND THEIR APPLICATIONS

1. INTRODUCTION

An extensive literature is presently available demonstrating the suitability of Petri nets to the modelling of discrete-event systems involving parallel processing. We refer to [PET 77] as introductory survey of most of this literature.

On the other hand theoretical studies are available [PET 76], [HACK], treating Petri nets from the viewpoint of automata and formal language theory.

In this Report an effort is made to combine these two approaches to Petri nets; namely, it provides an easy introduction to Petri nets and their languages and demonstrates the applicability of Petri net languages to the study of systems involving concurrency.

We assume the reader is familiar with the elementary concepts of formal language theory.

2. PETRI NETS AND THEIR LANGUAGES

PETRI GRAPHS AND PETRI NETS

Def. 2.1 A Petri Graph is a 3-tuple

$$G = (P, T, R)$$

where $P = \{p_1, \dots, p_n\}$ is a finite, nonempty set of places

$T = \{t_1, \dots, t_r\}$ is a finite, nonempty set of transitions


R is a binary relation on $P \cup T$ satisfying the condition

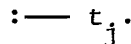
$$R \subseteq (P \times T) \cup (T \times P)$$

In the sequel ω will denote the set of non-negative integers.

A Petri Net N is a pair (G, m) where m is a marking of G , i.e. a function $m: P \rightarrow \omega$.

A Petri net N is conveniently represented in the following graphical form.

A place p_i is represented by a circle labelled by p_i : .

A transition t_j is represented by a bar labelled by t_j : .

The binary relation R is represented by directed edges from places to transitions or vice versa, in the usual way.

Finally, the integer $k = m(p_i)$ is written inside the circle representing the place p_i . Usually, one does not write 0 inside the circle and frequently the integer k is replaced by k dots (also called tokens).

Example 2.1 Let $N_1 = (P, T, R, m)$,

where $P = \{p_1, \dots, p_6\}$, $T = \{t_1, \dots, t_4\}$,

$R = \{(p_1, t_2), (p_2, t_1), (p_3, t_2), (p_4, t_3), (p_5, t_4), (p_6, t_3),$
 $(t_1, p_1), (t_2, p_2), (t_2, p_4), (t_3, p_3), (t_3, p_5), (t_4, p_6)\}$,

and $m(p_1) = m(p_4) = m(p_5) = 0$, $m(p_2) = m(p_6) = 1$, $m(p_3) = 5$.

The Petri net N_1 is represented by Fig. 2.1.

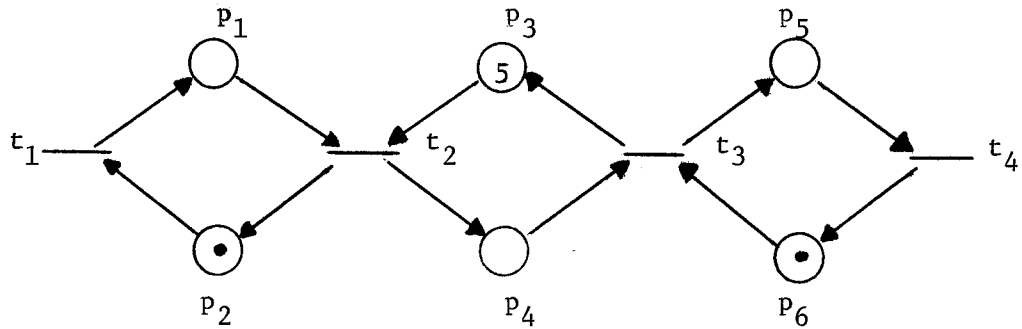


Fig. 2.1 The Petri net N_1 .

Example 2.2 Let $N_2 = (P, T, R, m)$,

where $P = \{p_1\}$, $T = \{t_1, t_2\}$, $R = \{(p_1, t_2), (t_1, p_1)\}$, and $m(p_1) = 0$.

The Petri net N_2 is shown in Fig. 2.2

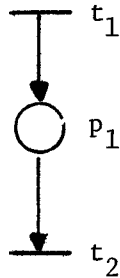


Fig. 2.2 The Petri net N_2 .

Example 2.3 Another Petri net N_3 , is shown in Fig. 2.3.

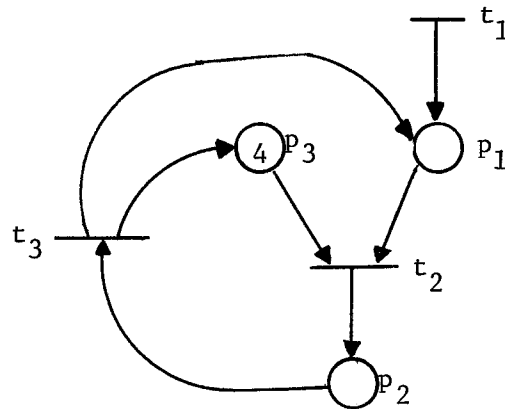


Fig. 2.3 The Petri net N_3 .

Petri nets are used to model dynamic systems, changing in time. In order to simulate such changes, so-called firing rules for Petri nets will now be defined. Such "firing" will not change the Petri graph G but may change the marking m into another marking m' which is again a function $m' : P \rightarrow \omega$. A further "firing" (in accordance with the given firing rules) may lead to a marking m'' , etc. Various markings of G correspond to various states of the dynamic system simulated by G with its markings.

FIRING RULES

Def. 2.2 The transition t_j of the Petri net $N = (G, m)$ is enabled iff every input place p_i of t_j , i.e. $p_i R t_j$, satisfies the condition $m(p_i) > 0$.

Note that according to this definition, a transition t_j which has no input places is always enabled. In each of the Petri nets N_1 , N_2 , and N_3 , (see Figs. 2.1, 2.2, 2.3) the transition t_1 only is enabled.

Def. 2.3 The firing of an enabled transition t_j consists of removing one token from each input place of t_j and adding one token to each output place of t_j (i.e. a place p_k such that $t_j R p_k$). More formally, the firing of t_j changes the marking m of G into the marking m' as follows:-

$$(1) \quad (\forall p_i \in P) \quad p_i R t_j \wedge \neg t_j R p_i \Rightarrow m'(p_i) = m(p_i) - 1$$

$$(2) \quad (\forall p_i \in P) \quad \neg p_i R t_j \wedge t_j R p_i \Rightarrow m'(p_i) = m(p_i) + 1$$

$$(3) \quad \text{For all other places } p_i \text{ (including those which are both an input and an output place of } t_j), m'(p_i) = m(p_i).$$

Notice that only enabled transitions may fire. We write

$m \xrightarrow{t_j} m'$ to state that the marking m' is obtained from the marking m as a result of firing the transition t_j .

Def. 2.4 Let $w \in T^+$, i.e. w is a finite string of transitions

$w = t_{j_1} t_{j_2} \dots t_{j_\ell}$. w is called a firing sequence of the Petri net

$N = (G, m)$ iff there exist markings m_1, \dots, m_ℓ such that

$$m \xrightarrow{t_{j_1}} m_1, m_1 \xrightarrow{t_{j_2}} m_2, \dots, m_{\ell-1} \xrightarrow{t_{j_\ell}} m_\ell.$$

In this case we write $m \xrightarrow{w} m_\ell$.

We now bring some examples of firings and firing sequences. It will be convenient to represent a marking $m : P \rightarrow \omega$ by the n -dimensional vector $(m(p_1), \dots, m(p_n))$. For the Petri net N_1 (Fig. 2.1), $t_1 t_2 t_1$ is a firing sequence. Indeed,

$$(0, 1, 5, 0, 0, 1) \xrightarrow{t_1} (1, 0, 5, 0, 0, 1) \xrightarrow{t_2} (0, 1, 4, 1, 0, 1) \xrightarrow{t_1} (1, 0, 4, 1, 0, 1).$$

Note that under the marking $(0, 1, 4, 1, 0, 1)$ both transitions t_1 and t_3 are enabled. Hence $t_1 t_2 t_3$ is also a firing sequence for N_1 . The following are examples of firing sequences for N_2 (see Fig. 2.2).

$$(0) \xrightarrow{t_1} (1) \xrightarrow{t_1} (2) \xrightarrow{t_1} (3) \xrightarrow{t_2} (2) \xrightarrow{t_1} (3)$$

$$(0) \xrightarrow{t_1} (1) \xrightarrow{t_2} (0) \xrightarrow{t_1} (1) \xrightarrow{t_2} (0)$$

For N_3 of Fig. 2.3 we have, for example,

$$(0, 0, 4) \xrightarrow{t_1} (1, 0, 4) \xrightarrow{t_1} (2, 0, 4) \xrightarrow{t_2} (1, 1, 3) \xrightarrow{t_3} (2, 0, 4)$$

$$(0, 0, 4) \xrightarrow{t_1} (1, 0, 4) \xrightarrow{t_2} (0, 1, 3) \xrightarrow{t_1} (1, 1, 3) \xrightarrow{t_2} (0, 2, 2)$$

Note that in a Petri net N more than one transition may be enabled simultaneously. We postulate that only one transition may fire at a time, but no priorities are assigned. Thus, starting with a given

marking, various firing sequences may occur.

One easily finds examples in which the firing of one enabled transition converts another enabled transition into a non-enabled transition. See e.g. Fig. 2.4.

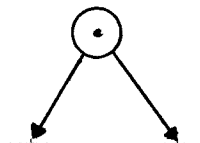


Fig. 2.4

AN ILLUSTRATIVE APPLICATION

Consider two cooperating processes, one called the PRODUCER and the other the CONSUMER [DIJ]. The PRODUCER may produce any quantity of a certain item. Each time the PRODUCER produces an item it deposits it in a store of finite capacity (M items), provided the store is not full. The CONSUMER may take an item from the store, provided the store is not empty, and consume it, before it is allowed to access the store again. The problem is to design a system which will coordinate these two processes. The Petri net N_1 of Fig. 2.1 represents such a system for $M=5$. The transitions t_1, t_2, t_3, t_4 correspond to the actions of producing, depositing, taking and consuming, respectively, and will be denoted in the sequel by p, d, t , and c .

Later on we shall provide a formal proof that any firing sequence of N_1 corresponds to a sequence of actions of the PRODUCER-CONSUMER system obeying the above rules, and viceversa: to every such "legal" sequence of actions corresponds a firing sequence of N_1 . For example, the following firing sequence of N_1

pdpdptctpcdpd

clearly represents a legal sequence of actions. The following observations may serve as an intuitive justification of our claim.

For any firing sequence of N_1 we must have:

- (1) The sequence must start with p and between any two occurrences of p there must be an occurrence of d and vice versa.
- (2) In the sequence t must appear before c and between any two occurrences of t there must be an occurrence of c and vice versa.
- (3) The number of t 's cannot exceed the number of d 's and the first occurrence of d must be before the first occurrence of t .
- (4) The number of d 's less the number of t 's must not exceed 5.

Note that any prefix of a firing sequence is itself a firing sequence and thus satisfies the above restrictions.

PETRI NET LANGUAGES

Evidently, not every word $w \in T^+$ is a firing sequence of a given Petri net N . Thus a Petri net can be used to define subsets of T^+ , i.e. languages over the alphabet T . Formally, we define

Def. 2.5 The set of all firing sequences of a Petri net N is called the language of N and will be denoted by $L(N)$.

For the Petri net N_a of Fig. 2.5(a) we have

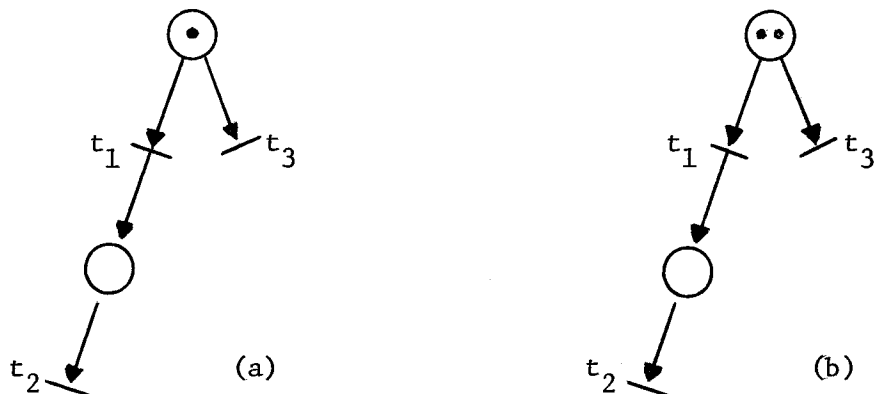


Fig. 2.5 (a) Petri net N_a (b) Petri net N_b

$L(N_a) = \{t_1, t_3, t_1t_2\}$. For the net of Fig. 2.5(b) we obtain

$$L(N_3) = \{t_1, t_3, t_1t_1, t_1t_2, t_1t_3, t_3t_3, t_1t_1t_2, t_1t_2t_1, t_1t_2t_3, t_1t_3t_2, \\ t_3t_1t_2, t_1t_1t_2t_2, t_1t_2t_1t_2\}.$$

The Petri net N of Fig. 2.6 describes two sequential processes A and B , operating concurrently, but independently. Process A consists of action t_1 , followed by action t_2 . Similarly, process B consists of action t_3 , followed by action t_4 .

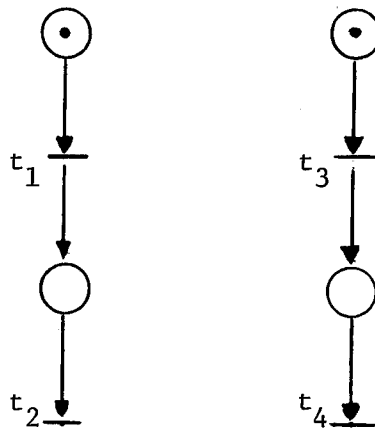


Fig. 2.6

The language of this Petri net, given below, represents all possible partial and complete action sequences, provided no two actions occur at the very same moment.

$$L(N) = \{t_1, t_3, t_1t_2, t_1t_3, t_3t_1, t_3t_4, t_1t_2t_3, t_1t_3t_2, t_1t_3t_4, t_3t_1t_2, \\ t_3t_1t_4, t_3t_4t_1, t_1t_2t_3t_4, t_1t_3t_2t_4, t_1t_3t_4t_2, t_3t_1t_2t_4, t_3t_1t_4t_2, \\ t_3t_4t_1t_2\}$$

The above three languages are finite and could thus be defined by finite automata. However, the automata defining the last two languages will have a comparatively large number of states.

Consider now the Petri net N of Fig. 2.7.

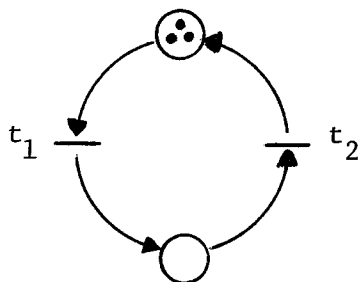


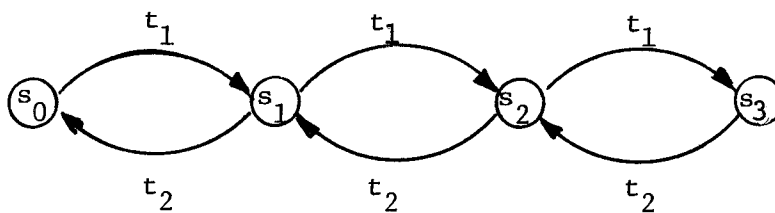
Fig. 2.7

Its language $L(N)$ can easily be characterized as follows

$$w \in L(N) \Leftrightarrow w \in \{t_1, t_2\}^+ \text{ and for every prefix } u \text{ of } w,$$

$$0 \leq \#(t_1, u) - \#(t_2, u) \leq 3$$

where $\#(t, w)$ denotes the number of occurrences of the letter t in the word w . It is easily seen (see Chapter 4) that the language $L(N)$ is regular. Indeed, it is defined by the finite automaton shown in Fig. 2.8.



s_0 is the initial state

All states are final.

Fig. 2.8

The above Petri net languages were all regular. However it can be shown that the language $L(N_2)$ (see Fig. 2.2) is not regular. $L(N_2)$ can be characterized as follows:-

$w \in L(N_2) \Leftrightarrow w \in \{t_1, t_2\}^+$ and for every prefix u of w , $\#(t_1, u) \geq \#(t_2, u)$.

This language is context-free. It is generated by the following context-free grammar:

$$S \rightarrow SS$$

$$S \rightarrow t_1 S t_2$$

$$S \rightarrow t_1 S$$

$$S \rightarrow t_1$$

$$S \rightarrow t_1 t_2$$

So far we have defined the language $L(N)$ for a given Petri net N as the set of all firing sequences of N . It is often useful to consider only such firing sequences which lead to a marking in a prescribed set F of final markings. Formally, we define for a given Petri net $N = (G, m)$ and a given set F of markings of G :

Def. 2.6 $L_F(N) = \{w \in T^+ \mid (\exists m' \in F) m \xrightarrow{w} m'\}$

E.g., consider the concurrent processes A and B represented in Fig. 2.6. If one wishes to represent only the complete action sequences, this can be done by means of the language $L_F(N)$, where N is the net of Fig. 2.6, and $F = \{(0,0,0,0)\}$. This language becomes

$$L_F(N) = \{t_1 t_2 t_3 t_4, t_1 t_3 t_2 t_4, t_1 t_3 t_4 t_2, t_3 t_1 t_2 t_4, \\ t_3 t_1 t_4 t_2, t_3 t_4 t_1 t_2\}.$$

As another example, consider the Petri net N_2 of Fig. 2.2 and let $F = \{(0)\}$. Then $L_F(N_2) = \{w \in L(N_2) \mid \#(t_1, w) = \#(t_2, w)\}$. This language (together with the empty string λ) is known as 1-type parantheses language or 1-type Dyck language and is well known to be context-free, but not regular [HO - UL].

Its context-free grammar is:

$$S \rightarrow SS$$

$$S \rightarrow t_1 S t_2$$

$$S \rightarrow t_1 t_2$$

We now introduce an additional way of associating languages with Petri nets. Namely, let $N = (P, T, R, m)$ be a Petri net, Σ a finite alphabet and η a mapping $\eta: T \rightarrow \Sigma \cup \{\lambda\}$ where λ denotes the empty string. The mapping η may be interpreted as a labelling of the transitions of N by letters from Σ and λ . Notice that more than one transition may be labelled by the same letter. It is customary to omit the label λ .

We call the triple $\Gamma = (N, \Sigma, \eta)$ a labelled Petri net and, define the languages $L(\Gamma)$ and $L_F(\Gamma)$ as follows.

Def. 2.7 $L(\Gamma) = \eta(L(N)) = \{\eta(w) \mid w \in L(N)\}$

$$L_F(\Gamma) = \eta(L_F(N)).$$

One may of course consider a Petri net N as a labelled Petri net with $\Sigma = T$ and η the identity mapping.

3. VECTOR ADDITION SYSTEMS

In this chapter we introduce the concept of Vector Addition System [KA - MI] and discuss some of its properties. These systems are very useful for the study of Petri nets.

Def. 3.1 An n-dimensional Vector Addition System (VAS) is an ordered pair

$$A = (V, z)$$

where V is a finite set of n -dimensional integer vectors and z is an n -dimensional vector of non-negative integers.

Recall that ω denotes the set of non-negative integers. For a given $x \in \omega^n$ and $v \in V$ such that $x' = x + v \in \omega^n$ we set

$$x \xrightarrow{v} x'.$$

Similarly to Def. 2.4, we now introduce the concept of a firing sequence in a VAS.

Def. 3.2 Let $w \in V^+$, i.e. w is a finite string of vectors $w = v_1 v_2 \dots v_\ell$. w is called a firing sequence of the VAS $A = (V, z)$ iff there exist vectors z_1, \dots, z_ℓ in ω^n such that

$$z \xrightarrow{v_1} z_1, z_1 \xrightarrow{v_2} z_2, \dots, z_{\ell-1} \xrightarrow{v_\ell} z_\ell.$$

In this case we write $z \xrightarrow{w} z_\ell$ as well as $z + w = z_\ell$. Clearly, if $z \xrightarrow{w} z_\ell$ then for every $k, 1 \leq k \leq \ell$, $z_k = z + v_1 + \dots + v_k \geq 0$

Example 3.1 Let A_1 be the 6-dimensional VAS $A_1 = (V, z)$

$$\text{where } V = \{v_1 = (+1, -1, 0, 0, 0, 0)$$

$$v_2 = (-1, +1, -1, +1, 0, 0)$$

$$v_3 = (0, 0, +1, -1, +1, -1)$$

$$v_4 = (0, 0, 0, 0, -1, +1)\}$$

and $z = (0,1,5,0,0,1)$.

$w = v_1 v_2 v_1$ is an example of a firing sequence in A_1 . Indeed

$$(0,1,5,0,0,1) \xrightarrow{v_1} (1,0,5,0,0,1) \xrightarrow{v_2} (0,1,4,1,0,1) \xrightarrow{v_1} (1,0,4,1,0,1).$$

Example 3.2 Let A_2 be the 3-dimensional VAS $A_2 = (V, z)$

where $V = \{v_1 = (-1,2,0), v_2 = (1,-3,2), v_3 = (0,0,-1)\}$

$$z = (4,0,1).$$

We have

$$(4,0,1) \xrightarrow{v_1} (3,2,1) \xrightarrow{v_3} (3,2,0) \xrightarrow{v_1} (2,4,0) \xrightarrow{v_2} (3,1,2).$$

Thus $w = v_1 v_3 v_1 v_2$ is a firing sequence of A_2 .

Evidently $v_1 v_3 v_2$ is not a firing sequence of A_2 .

Def. 3.3 The set of all firing sequences of a VAS A is called the language of A and will be denoted by $L(A)$.

Def. 3.4 The reachability set $R(A)$ of a VAS A is the set

$R(A) = \{x \mid (\exists w \in V^+) z \xrightarrow{w} x\} \cup \{z\}$, i.e. $R(A)$ consists of all vectors in ω^n "reachable" from z by applying to it a firing sequence of A and the vector z itself.

There exists an obvious relationship between Petri nets and VAS's. Namely, let $N = (P, T, R, m)$ be a Petri net and let us assume that N is selfloop-free, i.e. N does not contain a configuration as shown in Fig. 3.1.

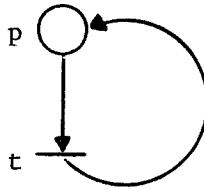


Fig. 3.1

Note that N is selfloop-free, iff $R \cap R^{-1} = \phi$. Let z be the n -dimensional vector $(m(p_1), \dots, m(p_n))$, and associate with every transition t_j the n -dimensional vector $v_j = (v_j[1], \dots, v_j[n])$, where

$$v_j[i] = \begin{cases} -1 & \text{if } p_i R t_j \\ +1 & \text{if } t_j R p_i \\ 0 & \text{otherwise} \end{cases}$$

In view of the above restriction of N , $v_j[i] = 0$ implies that no edge exists in either direction between p_i and t_j . Let V be the set of all v_j 's corresponding to transitions $t_j \in T$. The pair $A = (V, z)$ is a VAS, which corresponds to the Petri net N . Evidently, $L(A) = L(N)$, and $R(A)$ represents in the obvious way the set of all markings m' reachable from m by applying a firing sequence of N . The VAS A_1 of example 3.1 corresponds to the Petri net N_1 of Example 2.1.

It is noteworthy that the restriction to selfloop-free Petri net is essential in order to always have $L(A) = L(N)$. Indeed, the language of the Petri net N of Fig. 3.1 is the empty set. On the other hand, in the VAS A corresponding in a "natural" way to N , the vector v representing the transition t would be $v = (-1+1=0)$. Thus A would become $A = (V=\{v=(0)\}, z=(0))$, and its language $L(A)$ becomes $L(A) = v^+$.

Vector Addition Systems have applications beyond the representation of Petri nets (see [KA-MI]). For our purposes, we are especially interested in the question whether for a given VAS A , $R(A)$ is finite. The answer can be given using the following algorithm due to Karp and Miller [KA-MI]. Given an n -dimensional VAS $A = (V, z)$, we construct a corresponding rooted tree (A rooted tree is a connected, directed graph, in which every vertex except one, the root, has exactly one incoming edge.

The root has no incoming edge), labelled by vectors in ω^n .

- (1) The root is labelled by z .
- (2) If η is a vertex labelled by the vector $\ell(\eta)$, and there exists an ancestor ξ of η (i.e. there exists a directed path in the rooted tree from the vertex ξ to the vertex η), such that $\ell(\eta) \geq \ell(\xi)$, then η is a leaf of the rooted tree, i.e. has no outgoing edges. Otherwise, the immediate successors of η are in one-to-one correspondence with the non-negative vectors $\ell(\eta) + v_i$ ($v_i \in V$) and are labelled by these vectors.

Theorem 3.1 [KA-MI] (a) The above rooted tree is always finite.

(b) $R(A)$ is infinite iff there exists a leaf η such that $\ell(\eta) \geq \ell(\xi)$ for some ancestor ξ , but $\ell(\eta) \neq \ell(\xi)$.

Proof (a) Assume the tree is infinite. Since each vertex has a finite number of immediate successors, König's Infinity Lemma applies. Thus there exists an infinite, directed path η_1, η_2, \dots , in this tree. Now consider the infinite sequence of vectors $\ell(\eta_1), \ell(\eta_2), \dots$. Extract from this sequence an infinite subsequence nondecreasing in the first coordinate, extract from this subsequence an infinite subsequence of vectors, nondecreasing in the second coordinate, etc. till the last coordinate. Thus the above path contains two vertices η_i, η_j , such that $i < j$, i.e. η_i is an ancestor of η_j , and $\ell(\eta_i) \leq \ell(\eta_j)$. But this implies that η_j is a leaf. Hence the above infinite path cannot exist. Thus the tree must be finite.

(b) Assume there exists a leaf η such that $\ell(\eta) \geq \ell(\xi)$ for some ancestor ξ of η , but $\ell(\eta) \neq \ell(\xi)$. Clearly, in view of the above construction of the rooted tree $\ell(\xi) \in R(A)$ for every vertex ξ of the tree. Thus there exists a firing sequence w such that $z \xrightarrow{w} \ell(\xi)$. Since

ξ is an ancestor of η , there exists a word $x \in V^+$ such that $\ell(\xi) \xrightarrow{x} \ell(\eta)$. Since $\ell(\eta) \geq \ell(\xi)$, x must be applicable to $\ell(\eta)$ as well. Let $d = \ell(\eta) - \ell(\xi)$. Then $d \geq 0$, and $d \neq 0$. Clearly, the application of x to $\ell(\eta)$ yields the vector $\ell(\eta) + d$, to which x is again applicable, etc. Thus wx^k is a firing sequence of A for any $k = 0, 1, 2, \dots$ and the corresponding vectors $z + kd$ are elements of $R(A)$, all different. Thus $R(A)$ is infinite. Conversely, assume no leaf η exists in the tree, satisfying the above condition. It is obvious from the construction of the tree, that in this case the set $R(A)$ coincides with the set of labels of the tree, and thus $R(A)$ is finite. □

It should be emphasized that some vertex η may be a leaf of the tree due to the fact that no vector $v_i \in V$ is applicable to $\ell(\eta)$.

Evidently, the above algorithm can be terminated as soon as a leaf η is reached such that $\ell(\eta) \geq \ell(\xi)$, and $\ell(\eta) \neq \ell(\xi)$, for some ancestor ξ of η .

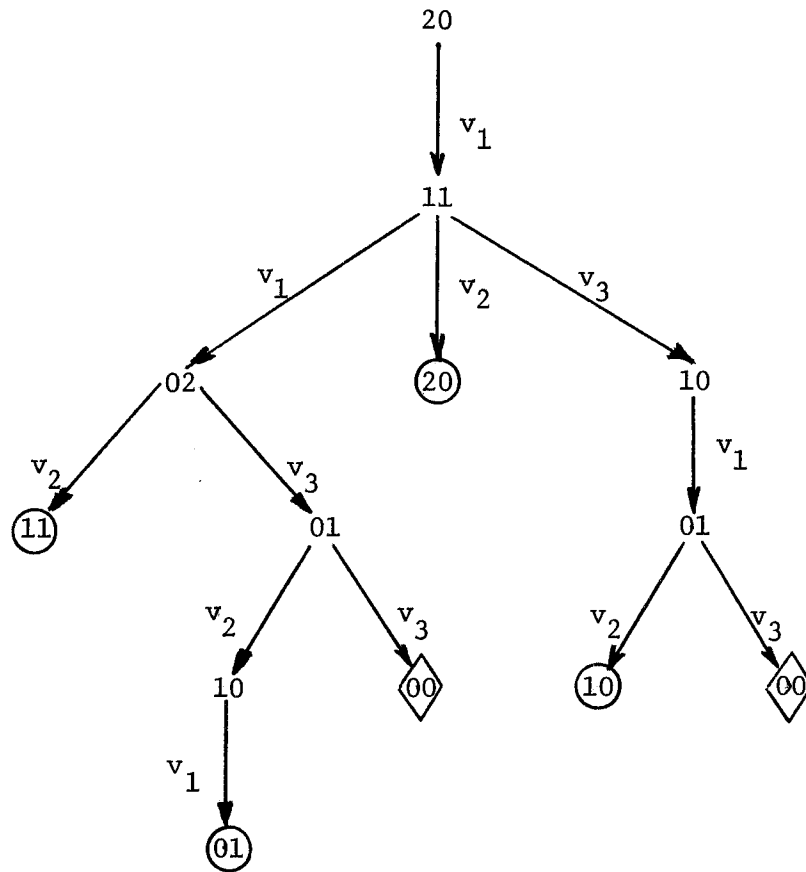
Example 3.3 Let $A = (V, z)$

where $V = \{v_1 = (-1, 1), v_2 = (1, -1), v_3 = (0, -1)\}$

and $z = (2, 0)$.

The corresponding rooted tree is shown overleaf. In the tree, leaves η such that $\ell(\eta) = \ell(\xi)$, for some ancestor ξ , are indicated by circles. Leaves to the labels of which no vector is applicable, are indicated by the rhombs. Since all leaves in the tree are only of these two types, $R(A)$ for this VAS is finite and we have

$$R(A) = \{20, 11, 02, 10, 01, 00\}.$$

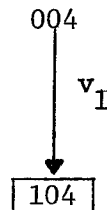


Example 3.4 Consider the VAS corresponding to the Petri net N_3 (see Fig. 2.3). Here

$$v_1 = (1,0,0), v_2 = (-1,1,-1), v_3 = (1,-1,1),$$

and $z = (0,0,4)$.

The rooted tree becomes:



In this tree, the leaf η with the label $\ell(\eta) = 104$ satisfies the condition $\ell(\eta) \geq \ell(\xi)$, $\ell(\eta) \neq \ell(\xi)$, where ξ is an ancestor of η . Such a

leaf is indicated by a square.

The set $R(A)$ in this example is infinite.

In the last example, the use of the above algorithm was very efficient. On the other hand, for the VAS $A = (V=\{(-1)\}, z=(10^6))$, it is immediately clear that $R(A)$ is finite, while the rooted tree obtained by the above algorithm will contain 10^6+1 vertices. Similarly the application of this algorithm to show that $R(A_1)$ for the VAS A_1 of Example 3.1 is finite, is quite tedious, while the following simple observation yields this result immediately.

Observation Let $A = (V, z)$ be a VAS.

If $(\forall v \in V) \sum_{i=1}^n v[i] \leq 0$, then $R(A)$ is finite.

Proof Indeed, let $z' \in R(A)$, i.e. there exists $w = v_{j_1} v_{j_2} \dots v_{j_\ell} \in V^+$ such that $z \xrightarrow{w} z'$.

Hence $z' = z + \sum_{k=1}^{\ell} v_{j_k}$ and

$$\begin{aligned} \sum_{i=1}^n z'[i] &= \sum_{i=1}^n z[i] + \sum_{i=1}^n \sum_{k=1}^{\ell} v_{j_k}[i] \\ &= \sum_{i=1}^n z[i] + \sum_{k=1}^{\ell} \sum_{i=1}^n v_{j_k}[i] \leq \sum_{i=1}^n z[i]. \end{aligned}$$

Clearly the set of non-negative vectors z' satisfying this inequality is finite. □

Sometimes the finiteness of the reachability set is easily recognized, even if the condition of the above observation does not hold in a straight-forward manner. The following VAS (V, z) describes a simplified version of a solution of the Mutual Exclusion Problem [DLJ] for two processes.

$$\begin{aligned}V &= \{v_1 = (-1, 1, 0, 0, -1) \\ &\quad v_2 = (1, -1, 0, 0, 1) \\ &\quad v_3 = (0, 0, -1, 1, -1) \\ &\quad v_4 = (0, 0, 1, -1, 1)\} \\ z &= (1, 0, 1, 0, 1).\end{aligned}$$

The condition of the above observation does not hold in view of v_2 and v_4 . Note, however, that the applications of v_1 and v_2 must alternate, with v_1 applied first. The same holds for v_3 and v_4 . But $v_1 + v_2 \leq 0$, and $v_3 + v_4 \leq 0$. Hence the argument of the above observation is again applicable and the reachability set of the VAS is finite.

4. PETRI NET LANGUAGES

In this chapter we give various characterizations of Petri net languages and study their relationship with regular languages.

We first introduce a notation which is very helpful for the description of Petri net languages.

Let Σ be a finite alphabet and w a word over Σ , i.e. $w \in \Sigma^*$.

$\text{pref}(w)$ denotes the set of all prefixes of w , i.e.

$$\text{pref}(w) \triangleq \{v \in \Sigma^+ \mid (\exists u \in \Sigma^*) vu = w\}.$$

$\#(\sigma, w)$ denotes the number of occurrences of the letter $\sigma \in \Sigma$ in w .

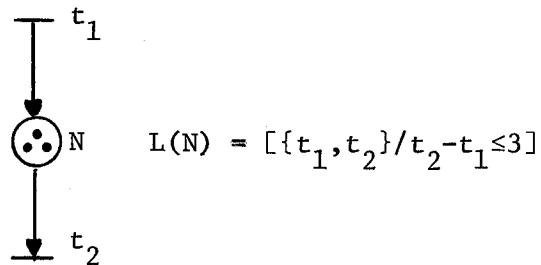
Let A and B be disjoint subsets of Σ , and $k \in \omega$. Then $[\Sigma/A-B \leq k]$ denotes the set of all words $w \in \Sigma^+$ such that in every prefix of w the number of occurrences of letters from A less the number of occurrences of letters from B is not less than k . In symbols,

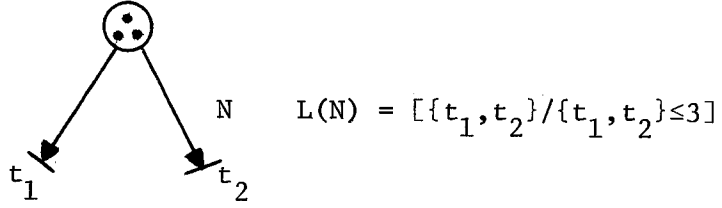
$$[\Sigma/A-B \leq k] \triangleq \{w \in \Sigma^+ \mid (\forall v \in \text{pref}(w)) \Delta(A, B, v) \leq k\}$$

$$\text{where } \Delta(A, B, v) \triangleq \sum_{\sigma \in A} \#(\sigma, v) - \sum_{\sigma \in B} \#(\sigma, v).$$

If $A = \{a\}$ or $B = \{b\}$, then the corresponding braces $\{ \}$ are usually omitted in the above symbols. We shall refer to languages of the above form as Δ -languages.

The following examples illustrate the application of the above notation.





The following result is essentially due to M. Hack [HACK].

Theorem 4.1 For every selfloop-free Petri net N , $L(N)$ can be described as an intersection of a finite number of Δ -languages.

Proof Let $N = (P, T, R, m)$, where $P = \{p_1, \dots, p_n\}$. With every place $p_i \in P$ we associate the Δ -language $L_i = [T / R(p_i) - R^{-1}(p_i) \leq m]p_i]$.

Notice that $R(p_i) \cap R^{-1}(p_i) = \phi$, since N is selfloop-free.

We claim that $L(N) = \bigcap_{i=1}^n L_i$.

Let $w \in L(N)$, and $m \xrightarrow{w} m'$. Then for any $p_i \in P$

$$m'(p_i) = m(p_i) + \Delta(R^{-1}(p_i), R(p_i), w).$$

Since $m'(p_i) \geq 0$, we have

$$\Delta(R(p_i), R^{-1}(p_i), w) \leq m(p_i).$$

Now $w \in L(N)$ implies $v \in L(N)$ for every prefix v of w . Thus the last inequality holds for every prefix v of w . Hence $w \in L_i$, and consequently

$$L(N) \subseteq \bigcap_{i=1}^n L_i.$$

Next, we show that $w \in \bigcap_{i=1}^n L_i$ implies $w \in L(N)$, using induction on the

length of w .

Thus, let $w = t \in T$, and assume $t \in L_i$ for every $1 \leq i \leq n$. We have to

show that t is enabled by m , i.e., $(\forall p_i \in R^{-1}(t)) m(p_i) \geq 1$. Indeed,

let $t \in R(p_i)$. Then $t \notin R^{-1}(p_i)$. Now $t \in L_i = [T/R(p_i) - R^{-1}(p_i) \leq m(p_i)]$

implies $1 - 0 \leq m(p_i)$, as required.

Let us now assume that $w \in \bigcap_{i=1}^n L_i \Rightarrow w \in L(N)$ for $w \in T^+$. We have to show

that for $t \in T$, $wt \in \bigcap_{i=1}^n L_i \Rightarrow wt \in L(N)$.

Indeed $wt \in L_i$ implies $w \in L_i$ since L_i contains together with a word all its prefixes. Hence, by our induction hypothesis $w \in L(N)$. Thus $m \xrightarrow{w} m'$, for some marking m' of $G = (P, T, R)$. We have to show that t is enabled by m' , i.e.

$$(\forall p_i \in R^{-1}(t)) m'(p_i) \geq 1$$

Notice first that $p_i R t$ implies

$$\sum_{\sigma \in R^{-1}(p_i)} \#(\sigma, wt) = \sum_{\sigma \in R^{-1}(p_i)} \#(\sigma, w),$$

$$\sum_{\sigma \in R(p_i)} \#(\sigma, wt) = \sum_{\sigma \in R(p_i)} \#(\sigma, w) + 1,$$

and

$$m'(p_i) = m(p_i) + \Delta(R^{-1}(p_i), R(p_i), w).$$

The assumption $wt \in L_i$ implies

$$\Delta(R(p_i), R^{-1}(p_i), wt) \leq m(p_i).$$

Thus $\sum_{\sigma \in R(p_i)} \#(\sigma, w) + 1 - \sum_{\sigma \in R^{-1}(p_i)} \#(\sigma, w) \leq m(p_i)$

Hence $m'(p_i) \geq 1$, as required. □

Example We now apply Theorem 4.1 to the Petri net N_1 of Fig. 2.1, replacing the transition labels t_1, t_2, t_3, t_4 by p, d, t, c , respectively. As discussed earlier (see Chapter 2) this net represents a solution to the PRODUCER-CONSUMER problem. Let $\Sigma = \{p, d, t, c\}$. We have

$$L_1 = [\Sigma / d-p \leq 0]$$

$$L_2 = [\Sigma / p-d \leq 1]$$

$$L_3 = [\Sigma / d-t \leq 5]$$

$$L_4 = [\Sigma / t-d \leq 0]$$

$$L_5 = [\Sigma / c-t \leq 0]$$

$$L_6 = [\Sigma / t-c \leq 1]$$

$$\text{and } L(N_1) = \bigcap_{i=1}^6 L_i$$

The above six languages correspond to the following six rules which jointly define the PRODUCER-CONSUMER problem.

- (1) L_1 - The number of deposits does not exceed the number of productions.
- (2) L_2 - The number of productions may not exceed the number of deposits by more than 1.
- (3) L_3 - The number of deposits may not exceed the number of takes by more than 5.
- (4) L_4 - The number of takes may not exceed the number of deposits.
- (5) L_5 - The number of consumptions may not exceed the number of takes.
- (6) L_6 - The number of takes may not exceed the number of consumptions by more than 1.

A word is in $L(N_1)$ iff it satisfies these six rules simultaneously.

This example demonstrates the suitability of Δ -languages to both the formulation of problems concerning concurrent processes and the

verification of their solutions.

Frequently it is convenient to describe Petri net languages by means of the so-called shuffle operation, which we denote by $||$.

Def. 4.1 Let Σ be a finite alphabet, and $x \in \Sigma^*$, $y \in \Sigma^*$. The shuffle $x||y \subseteq \Sigma^*$ is defined recursively as follows.

- (1) $\lambda||\lambda = \{\lambda\}$
- (2) For every $\sigma \in \Sigma$, $\sigma||\lambda = \lambda||\sigma = \{\sigma\}$
- (3) Let $\sigma \in \Sigma$, $\tau \in \Sigma$, $x \in \Sigma^*$, $y \in \Sigma^*$.

Then $\sigma x||\tau y = \{\sigma\} \cdot (x||\tau y) \cup \{\tau\} \cdot (\sigma x||y)$.

In words, if $x = \sigma_1\sigma_2\dots\sigma_n$ and $y = \tau_1\tau_2\dots\tau_k$, then $x||y$ is the set of all strings of length $n+k$, such that all the σ_i 's and τ_j 's appear in them exactly once, with the only restriction that the relative ordering of the σ_i 's is the same as in x , and the relative ordering of the τ_j 's is the same as in y . For example,

$$\begin{aligned} abc||de = \{abcde, abdce, abdec, adbce, adbec, \\ adeb, dabce, dabec, daebc, deabc\} \end{aligned}$$

For subsets $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$ we set

$$A||B = \{x||y \mid x \in A \wedge y \in B\}$$

Let $N_1 = (P_1, T_1, R_1, m_1)$ and $N_2 = (P_2, T_2, R_2, m_2)$ be two Petri nets such that $P_1 \cap P_2 = T_1 \cap T_2 = \phi$. Consider the Petri net $N = (P_1 \cup P_2, T_1 \cup T_2, R_1 \cup R_2, m_1 \cup m_2)$.

It is easily seen that

$$L(N) = [L(N_1) || L(N_2)] \cup L(N_1) \cup L(N_2).$$

For example, consider the Petri net of Fig. 2.6, and let N_1 and N_2 denote its two disjoint parts. Then $L(N_1) = \{t_1, t_1t_2\}$ and $L(N_2) = \{t_3, t_3t_4\}$.

One easily verifies that the language $L(N)$ for this net (see p. 2.7)

satisfies the above equality.

It follows immediately from Def. 4.1 that the shuffle operation is commutative and associative. In order to describe languages $L_F(N)$ (see Def. 2.6), we introduce the following notation:

$$[\Sigma / A-B = k] = \{w \in \Sigma^+ \mid \Delta(A,B,w) = k\}.$$

Let $N = (G,m)$ be a selfloop-free Petri net and let $F = \{m'\}$.

We associate with every place p_i of N the language

$$\bar{L}_i = [T / R(p_i) - R^{-1}(p_i) = m(p_i) - m'(p_i)]$$

Then one obtains

$$L_{\{m'\}}(N) = \bigcap_{i=1}^n (L_i \cap \bar{L}_i)$$

In general, we have

Theorem 4.2 For every selfloop-free Petri net $N = (P,T,R,m)$ and every set F of final markings of $G = (P,T,R)$,

$$L_F(N) = \bigcup_{m' \in F} L_{\{m'\}}(N).$$

In view of Def. 2.7, the language $L(\Gamma)$ of a labelled, selfloop-free Petri net $\Gamma = (N,\Sigma,\eta)$ can be obtained by applying the mapping η to the intersection of Δ -languages representing $L(N)$.

Furthermore, we have

Theorem 4.3 Let L be the intersection of a finite number of Δ -languages over a given alphabet Σ . Then there exists a selfloop-free Petri net N such that $L = L(N)$.

Proof Let $L = \bigcap_{i=1}^r L_i$ where

$$L_i = [\Sigma / A_i - B_i \leq k_i].$$

Let $N = (P, \Sigma, R, m)$, where

$$P = \{p_1, \dots, p_{r+1}\} \text{ and for every } i, 1 \leq i \leq r,$$

$$R(p_i) = A_i, R^{-1}(p_i) = B_i, \text{ and } m(p_i) = k_i.$$

Also, $R^{-1}(p_{r+1}) = \Sigma$, $R(p_{r+1}) = \phi$ and $m(p_{r+1}) = 0$.

By Theorem 4.1 we have

$$L(N) = \bigcap_{i=1}^n L_i \cap \Sigma^* = L.$$

□

The Petri nets considered in Theorem 4.1 were assumed to be selfloop-free. Indeed, consider the Petri net N of Fig. 4.1, which contains a selfloop.

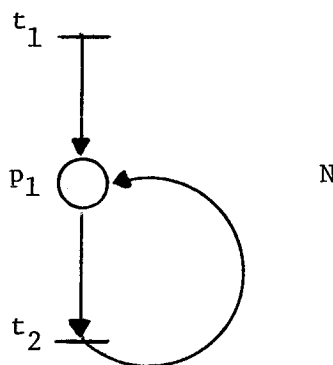


Fig. 4.1

Evidently, $L(N) = t_1 T^*$. However, this language cannot be obtained as intersection of Δ -languages. Indeed, a Δ -language is of the form $[\Sigma / A-B \leq k]$. In this case, it is useless to let Σ contain letters not in T , since $T \subseteq \Sigma$ implies $[\Sigma / A-B \leq k] \cap T^* = [T / A \cap T - B \cap T \leq k]$.

Hence we have to consider only Δ -languages of the following forms:

$$L_1 = [T/t_1 \leq k], L_2 = [T/t_2 \leq k], L_3 = [T/T \leq k],$$

$$L_4 = [T/t_1 - t_2 \leq k], L_5 = [T/t_2 - t_1 \leq k].$$

Evidently, $L(N)$ cannot be contained in any language of the above forms.

This example shows that the restriction of Theorem 4.1 to self-loop-free Petri nets is essential.

On the other hand, the above language $L(N)$ can be obtained by applying the mapping $\eta = \{(t_1, t_1), (t_2, t_2), (t_3, \lambda)\}$ to the language $L(N')$, where N' is the selfloop-free Petri net of Fig. 4.2.

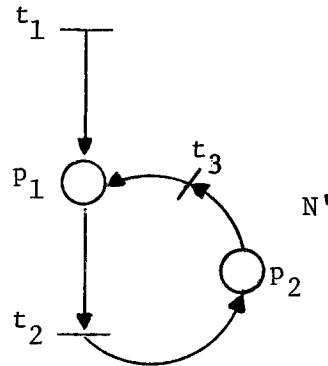


Fig. 4.2 Petri net N'

Indeed, let $w \in L(N)$. Replace in w every appearance of t_2 by $t_2 t_3$. The word obtained is clearly a firing sequence w' of N' and $\eta(w') = w$. Conversely $w' \in L(N')$ must begin with t_1 . Hence $\eta(w') \in t_1 \{t_1, t_2\}^* = L(N)$.

By applying the same argument to every selfloop of an arbitrary Petri net, we obtain

Theorem 4.4 For every Petri net N , $L(N)$ can be described as a homomorphic image of an intersection of a finite number of Δ -languages.

PETRI NET LANGUAGES AND REGULAR EXPRESSIONS

We have already discussed various examples of Petri nets the languages of which were regular expressions. Moreover, we have

Theorem 4.5 For every regular language L_R there exists a labelled Petri net Γ with a set F of final markings such that $L_F(\Gamma) = L_R$.

Proof Let $A = (Q, \Sigma, \delta, q_0, F_A)$ be the finite automaton recognizing L_R .

We define $\Gamma = (P, T, R, m, \Sigma, \eta)$ in the following way:

$P = Q$; for each transition in A $\delta(q, \sigma) = q'$ introduce a transition $t \in T$ and set qRt , tRq' , and $\eta(t) = \sigma$. Define m by $m(q_0) = 1$ and $m(q) = 0$ for every other $q \in P$; finally associate with every $q_F \in F_A$ a final marking $m' \in F$ such that $m'(q_F) = 1$ and $m'(q) = 0$ for every other $q \in P$. It is evident from this construction that the automaton A is precisely simulated by Γ , thus $L_F(\Gamma) = L(A) = L_R$. □

Consider now a labelled Petri net $\Gamma = (N, \Sigma, \eta)$, where $N = (P, T, R, m)$, together with a set of final markings F . Let $M(N)$ be the set of all markings of (P, T, R) reachable from m ; i.e.

$$M(N) = \{m' \mid (\exists w \in L(N)) m \xrightarrow{w} m'\}.$$

If $M(N)$ is finite, then $L_F(\Gamma)$ is regular. Indeed, Γ may be simulated by a finite, non-deterministic automaton A , the state set Q of which is $M(N)$. Whenever $m_1 \xrightarrow{t} m_2$ and $\eta(t) = \sigma$ in Γ , let $m_2 \in \delta(m_1, \sigma)$. Furthermore, $q_0 = m$, and $F_A \subseteq Q$ is the set $F_A = F \cap M(N)$. Then $L(A) = L_F(\Gamma)$.

An easy application of the above results is the following.

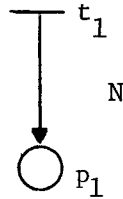
Theorem 4.6 Let L_1 and L_2 be regular languages. Then $L_1 \parallel L_2$ is regular.

Proof Let Γ_1 and Γ_2 be the labelled Petri nets generating L_1 and L_2 respectively, and let $\Gamma = \Gamma_1 \cup \Gamma_2$. Then $L(\Gamma) = L_1 \parallel L_2$. Now the set of markings reachable from Γ_i ($i=1,2$) is finite. Since Γ_1 and Γ_2 are both disjoint parts of Γ , the set of markings reachable from Γ is also finite.

Thus $L(\Gamma)$ is regular. □

For a proof of this result without the use of Petri nets, see [EIL].

The finiteness of $M(N)$ is a sufficient but not necessary condition for $L_F(\Gamma)$ to be regular. For example, the Petri net



defines the regular language $L(N) = \{t_1\}^+$. However $M(N)$ is evidently infinite. Fig. 4.1 provides an additional example of this type.

In [GI-YO] necessary and sufficient conditions are established for Vector Addition Systems to define regular languages, and an algorithm is designed to decide whether these conditions are satisfied. These results are, of course, immediately applicable to Petri nets.

REFERENCES

- [DIJ] E.W. Dijkstra, Cooperating sequential processes, in:
F. Genuys (ed.), Programming Languages, Academic Press, 1968,
pp. 43-112.
- [EIL] S. Eilenberg, Automata, Languages and Machines, Vol. A,
Academic Press, 1974.
- [GI-YO] A. Ginzburg and M. Yoeli, Vector addition systems and
regular languages, Research Report CS-78-43, Dept. of Comp.
Sci., Univ. of Waterloo, October 1978.
- [HACK] M. Hack, Petri net languages, Tech. Report 159, Lab. for
Comp. Sci., Mass. Inst. Tech., March 1976.
- [HO-UL] J.E. Hopcroft and J.D. Ullman, Formal Languages and their
Relation to Automata, Addison-Wesley, 1969.
- [KA-MI] R.M. Karp and R.E. Miller, Parallel program schemata,
J. Comp. Syst. Sci., 3, (1969), pp. 147-195.
- [PET 76] J.L. Peterson, Computation sequence sets, J. Comp. Syst. Sci.,
13 (1976), pp. 1-24.
- [PET 77] J.L. Peterson, Petri nets, Computing Surveys, 9, (Sept. 1977),
pp. 223-250.