LANGUAGE THEORY OF PETRI NETS

Matthias Jantzen

Fachbereich Informatik
Universität Hamburg
Rothenbaumchaussee 67
D-2000 Hamburg 13

ABSTRACT Petri nets where multiple arcs are allows and the capacity of the places need not be bounded are here called Place/Transition systems. The restrictions of the possible finite or infinite occurence sequences of a P/T-system to the transitions are called transition sequences and give the basis to define families of formal languages related to classes of P/T-systems.

We introduce the notation and give a survey on methods and results about sets of finite transition sequences. We will compare the classes of Petri net languages we obtain with other families of languages known from automata and formal language theory. We hope to convince that these techniques and results are useful for the formulation and solution of certain questions about P/T-systems, as well as for comparing the underlying systems.

Key words: Petri net language, firing sequence, closure properties, multicounter languages, intersection closed semi-AFL.

CONTENTS

Abstract
Introduction
Basic Notation and Definitions
Comparing Petri Net Languages
Closure Properties
Comparison with other Families
Specific Properties

INTRODUCTION

If one wants to analyze and derive properties of a system modelled by some Petri net, then this requires a great deal of formalism and a decision on which notion is most convenient or adequate. Here we consider P/T-systems - often called Petri nets - as descriptions of the systems and their possible transition sequences, i.e., the sequence of actions a single observer can establish, as partial descriptions of their behaviours.

Many questions about a system can be posed in terms of the sequences of actions that occur and thus may be answered by analyzing the underlying Petri net language. For instance, if a new and better system is designed to replace an existing one, then of course its behaviour should be identical to that of the old system. This can only happen if the languages of the underlying Petri net models are identical. Thus, in order to prove inequivalence of the two systems it is sufficient to exhibit one transition sequence that does not occur in both Petri net languages.

Transition sequences are surely not adequate to fully describe all aspects of a concurrent system and therefore trace languages [Mazurkiewicz 77], partial languages [Grabowski 79 b], semi-languages [Starke 81], and subset languages [Rozenberg/ Verraedt 83] have been defined. Also infinite transition sequences offer a possibility to define and study properties of systems like fairness and starvation.

All these approaches are discussed in several contributions to this volume and we therefore concentrate on the presentation of the traditional approach to study Petri net languages that began with the work of H. Baker [Baker 72], M. Hack [Hack 75] and J.L. Peterson [Peterson 74] and was continued by T. Etzion, A. Ginzburg, M. Jantzen, P.H. Starke, R. Valk, G. Vidal-Naquet, M. Yoeli and others.

The advantages of using the string language approach are based upon the following observations: With the help of Petri net languages we can classify and easily distinguish syntactically defined classes of Petri nets, and thus the underlying systems, by the classes of languages they can define. In some cases, properties of the reachability sets can be deduced from the transition sequences.

The methods used in formal language and automata theory which are applicable here sometimes provide simpler proofs for expected results and allow to formalize some of the folk-theorems. Also, most proofs are constructive and as such provide constructions and design principles. This remark applies especially to closure properties of Petri net languages that are proved by transforming the underlying nets. On the other hand, characterizations of Petri net languages via closure operations open the window to a direct comparison with other families of languages studied in theoretical computer science. For instance, the usual L-type Petri net languages are easily identified as the class of languages acceptable by one-way multi-counter machines in quasi-realtime while testing each counter for zero only a constant number of times [Greibach 78, Jantzen 79c].

The possibility of hiding or identifying transitions by their labelling opens more descriptive power and allows simpler formulations. Also this is convenient for a direct comparison with other classes of languages.

In what follows we basically use the notation from [Peterson 81] with slight variations (we use vectors instead of bags) and extensions according to [Best/Fernandez 86] and sometimes [Starke 80]. Also, because of lack of space, we do not give full proofs but only hints and the references where these can be found.

BASIC NOTATION AND DEFINITION

Looking at a P/T-system as a single observer it is impossible to record the concurrence of distinct transitions at the very same time and thus the sequence of transitions that occurred is the then basic object of consideration. As described in [Reisig 86] a transition sequence is obtained from an occurrence sequence by omitting all the markings reached. Such a transition sequence is free of labels and the set of all such transition sequences which are enabled at the initial marking of some P/T-system N is usually called a free (P-type) Petri net language. If one is interested in all transition sequences that change the initial marking of N into a final marking then there are various ways of how a final marking as the representative of the distributed state of the system to be reached should be specified.

It can be one marking out of a prescribed set (usually finite) of final markings that must be reached exactly (L-type) or it may suffice if finally a certain marking (or one of a prescribed set) is covered (G-type). This can be an adequate definition if the markings of the places, for instance, describe the number of resources available. A third possibility then could be to collect all transition sequences that yield a dead marking, i.e., one where no transition is enabled (T-type). All these types of languages have been studied by Peterson and Hack and can be equipped with a labelling of some sort. Restricting the otherwise nondeterministic choices of transition occurrences within labelled Petri nets one is lead to the definition of deterministic Petri nets [Vidal-Naquet 81, Pelz 86] which is similar to the notion of determinism as used for usual types of automata. In order to give the formal definitions of these languages and the classes (or families) they belong to we want to include some preliminary notion which is taken from standard text books on automata theory and formal languages and necessary in this context.

Notation

Let X be an alphabet then $X^* := X^+ \cup \{\lambda\}$ denotes the free monoid generated by X, where X^+ is the set of all finite non-empty strings (or words) over X, i.e., finite sequences of symbols from X juxtaposed. The transitive monoid operation in X^* is the concatenation and its application to the two strings u and v yields the string uv. The neutral element of X^* is called the empty string noted by λ. Given a string w and a symbol x then the non-negative integer $|w|_x \in \mathbb{N}$ denotes the number of occurrences of the symbol x within w, whereas $|w|$ is the total length of the string w. Hence for $w \in X^*$ we have $|w| = \sum_{x \in X} |w|_x$. The vector $\psi(w) := (|w|_{x_1}, |w|_{x_2}, \ldots, |w|_{x_n})$ is called Parikh vector of w, where $X = \{w_1, x_2, \ldots, x_n\}$ is ordered in the indicated way.

The homomorphism $\psi : X^* \longrightarrow \mathbb{N}^X$ where $w \longmapsto \psi(w)$ as defined above is called the Parikh mapping. Here, for sets A and B, B^A denotes the set of all total mappings from A to B and if A is an (ordered) finite set any such mapping can be denoted by a vector over B. Usually we use column-vectors but do write them as row vectors for a simpler lay-out. The difference is only important in connection with matrix products and then will become clear from the context. An alternative notation would use multisets or bags and is not done here.

In addition to the definitions in [Best + Fernandez 86] let us define column vectors $t^-, t^+ \in \mathbb{N}^S$ for each transition $t \in T$ by: $t^-(p) := W(p,t)$ and $t^+(p) := W(t,p)$ for all $p \in S$. Recall that $W(x,y)=0$ in case $(x,y) \notin F$. This notation from [Starke 78] is used to define a homomorphism $\Delta:T^* \longrightarrow \mathbb{Z}^S$ by $\Delta(t):=t^+-t^-$ which gives the t-column $C(\cdot,t)$ of the incidence matrix $C(s,t):=W(t,s)-W(s,t)$ for all $s\in S, t\in T$. The $|S|\times|T|$-matrices composed out of all the $t^-(t^+)$ column vectors are often called foreward (backward, resp.) incidence function [Hack 75], Prê (Post, resp.) matrices [Brams 83], and correspond to the input (output, resp.) functions in [Peterson 81]. With this definition of Δ we get for each finite transition sequence $w \in T^*$ the following equality: $\Delta(w)=C\cdot\psi(w)$. The enabling rule for t at m now becomes: $t^- \leqq m \leqq K - t^+$ and $m[w>m'$ implies $m'=m+\Delta(w)$.

At first, the transition rule [t> as defined in [Best/Fernandez 86] has to be generalized to transition sequences.

<u>Definition</u>

Let $N=(S,T,F,K,W,m)$ be a P/T-system. We then write:

(a) $m[\lambda>m$ for any marking $m \in \mathbb{N}^S$

(b) $m[w>m''$ if $w=ut$ for some $u \in T^*$, $t \in T$ and there exists $m' \in \mathbb{N}^S$ such that $m[u>m'[t>m''$.

(c) A transition sequence $w \in T^*$ is enabled at m, written m[w>, if there exists some m' such that m[w>m' holds. Specifically, this means that the empty transition sequence λ is always enabled .

Any subset $L \subseteq X^*$ is a (formal) <u>language</u> and for free Petri net languages the alphabet used will be the set T of transitions.

However, we also want to deal with whole families of languages and notice that a <u>family of languages</u> $\mathcal{L}$ is a nonempty set of languages which is closed under isomorphisms, i.e. change of alphabets, such that not all elements $L \in \mathcal{L}$ are the empty set. The first important family of Petri net languages, extensively studied in [Starke 78], is the basis for all other language families we consider. It is the class of <u>free Petri net languages</u> with or without final markings.

In order to change the initial marking of a P/T-system without changing the underlying P/T-net structure we use the following notation:

<u>Notation</u>

For any P/T-system $\Sigma=(S,T,F,K,W,m)$ the structure (S,T,F,K,W), or (S,T,F,W) if $K(p)=\infty$ for all $p \in S$, is called (the underlying) P/T-net.

In order to define a Petri net language one has to specify four quantities: the underlying Petri net $N=(S,T,F,K,W)$, the initial marking $m\in\mathbb{N}^S$, a usually finite set of final markings $M\subset\mathbb{N}^S$, and a labelling function $\ell:T\to X$, where X is an alphabet. Given this information $\Gamma:=(N,m,M,\ell)$ and by extending ℓ to a homomorphism $\ell:T^*\to X^*$ in the obvious way or omitting ℓ if it is the identity, we define the following types of Petri net languages and the corresponding language families.

Definition

Let $\Gamma=(N,m,M,\ell)$ as described above then

(a) $L(\Gamma):=\{\ell(w)\,|\,\exists w\in T^*\ \exists m'\in M\colon\ m[w>m'\}$ is the L-type or terminal language of N

(b) $G(\Gamma):=\{\ell(w)\,|\,\exists w\in T^*\ \exists m'\in M\ \exists m''\geq m'\colon\ m[w>m''\}$ is the G-type or covering language of N, also called weak language.

(c) $P(\Gamma):=\{\ell(w)\,|\,\exists w\in T^*\colon\ m[w>\}$ is the P-type language or the set of all labelled transition sequences of N.

(d) $T(\Gamma):=\{\ell(w)\,|\,\exists w\in T^*\ \exists m'\in\mathbb{N}^S\colon\ (m[w>m' \text{ and } \forall t\in T\colon\ m'[t>)\}$ is the T-type or deadlock language of N.

The families of L-type (G-type, P,type, T-type, resp.) languages are abbreviated by $\underline{L}$ ($\underline{G}$,$\underline{P}$,$\underline{T}$, resp.).

Among the many possiblities for choosing certain labelling functions, the most simple is the one where $|T|=|X|$ and ℓ is an isomorphism and may thus be omitted completely by choosing $X:=T$. In this situation the languages of either type are called free Petri net languages. A situation worth introducing an extra notation is the one where the labelling function is extended by allowing the empty word λ as a label of the transitions in order to hide them. Languages obtained from Petri nets this way are called arbitrary Petri net languages or languages defined by using so-called λ-transitions.

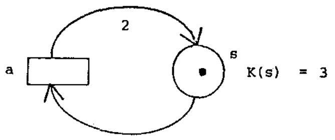
Definition

For any family X of Petri net languages of either type $X\in\{\underline{L},\underline{G},\underline{P},\underline{T}\}$ let X^f resp. X^λ denote the corresponding class of free resp. arbitrary Petri net languages.

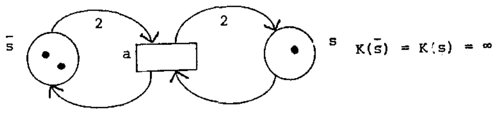
If we are dealing with capacities that restrict the concession of a transition this obviously changes the free Petri net language. On the other hand, places that are bounded in advance by their prescribed capacity can be modelled by introducing so-called complementary places. The construction is given in [Reisig 85] and allows to consider only P/T-systems without capacities since this construction does not introduce new transitions and thus does not change the language generated. In what follows we thus omit the capacity constraints completely.

Example

Consider the structure $\Gamma=(N,m,\emptyset,\sigma)$ where N,m and σ are displayed in the figure. Then $P(\Gamma)=\{\lambda,a\}$ if the capacity of s is given by $K(s):=3$ and the enabling rule of the transition is that defined in [Best/Fernandez 86]. If $K(s)=\infty$, then $P(\Gamma)=\{a\}^*$.



The labelled P/T-system simulating Γ with unbounded capacities and complementary place $\bar{s}$ and initial marking as displayed looks as follows:



It should be clear from the definition that any P-type Petri net language L has the <u>prefix property,</u> which means that L= Pref(L):={u∈T*|uv∈L} . This is no longer the case if we introduce final markings to be reached, covered or being a dead end.

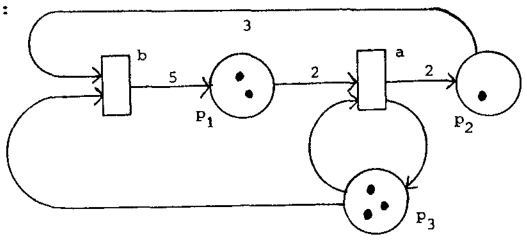
Also the Petri net language, even if it is free of labels, by no means gives enough information to recover the Petri net it came from, unless the Petri nets are selfloop-free and the languages are free ones. However, this is not - and has historically never been - the aim for studying Petri net languages.

The following example shows a Petri net where its free terminal language is just a single word but the free P-type language is larger than just the set of all its prefixes.

<u>Example</u>

The structure Γ consists of the following Petri net N together with initial marking m=(2,1,3) and a single final marking m'=(5,4,0) as indicated. One obtaines $L(N,m,m')=\{aba^2ba^3b\}\neq P(N,m,\emptyset)=G(N,m,\underline{0})=Pref(aba^2bab,aba^2ba^2b,aba^2ba^3b)$.

N:



By extending the automata theoretic notion of deterministic machines to Petri nets G. Vidal-Naquet introduced deterministic Petri nets:

<u>Definition</u>

A labelled Petri Net N=(S,T,F,W) with initial marking m_o , i.e., the structure $\Gamma=(N,m_o,\ell)$, is called deterministic if ∀t,t'∈T ∀m∈(m$_o$>: m(t> and m(t'> implies ($\ell(t)\neq\ell(t')$ or t=t') and ($\ell(t)=\lambda$ implies t=t') .

Definition

If $\Gamma=(N,m_o,\ell)$ is deterministic then $L(\Gamma)(T(\Gamma),G(\Gamma),P(\Gamma)$, resp.) are deterministic Petri net languages of the corresponding type.

For any family $X\in\{\underline{L},\underline{T},\underline{G},\underline{P}\}$ X_{det} (or $X^{\lambda}_{det},X^{f}_{det}$) denotes the family obtained by using deterministic Petri nets as the underlying structure where non-erasing (arbitrary, injective, resp.) labelling homomorphisms are used.

COMPARING PETRI NET LANGUAGES

Most of the results concerning free Petri net languages are contained in [Starke 78] where Starke corrects a statement by Hack [Hack 75]. The classes $\underline{G}^f$ and $\underline{T}^f$ are not so well studied so far, mostly because the free Petri net languages do not have very nice closure properties and closure operations are an important tool to compare families of languages and are usually considered in formal language theory.

Notice that any P-type language can be written as an G-type language by using the structure $(N,m,\underline{0})$ instead of (N,m) since any marking reachable from m exceeds the zero marking $\underline{0}$ of appropriate size. The preceding example already used this.

It is also not difficult to verify that the covering languages are also terminal Petri net languages which , however, are usually not free. The construction adds new transitions to the old structure in the following way: For each transition t one adds transitions $t_i, 1\leq i\leq k$, where $t_i^+\leq t^+$ and each such possibility has to be used, which determines k to the product of all entries in t^+ .

The following table of basic inclusions if from [Peterson 81] and mainly follows from considerations similar to those above. An exception is the proof of $\underline{T}=\underline{L}$ from [Parigot/Pelz 85] which is not contained in Peterson's work.

Theorem

The following inclusions (in general not strict) are valid for the families of Petri net languages defined so far.

$$\begin{array}{ccccc}
\underline{L}^{\lambda} & \supset & \underline{L} & \supset & \underline{L}^{f} \\
\| & & \| & & \\
\underline{T}^{\lambda} & \supset & \underline{T} & \supset & \underline{T}^{f} \\
\cup & & \cup & & \\
\underline{G}^{\lambda} & \supset & \underline{G} & \supset & \underline{G}^{f} \\
\cup & & \cup & & \cup \\
\underline{P}^{\lambda} & \supset & \underline{P} & \supset & \underline{P}^{f}
\end{array}$$

A first comparison of some deterministic Petri net languages was given in [Vidal-Naquet 81] and yielded the following strict inclusions.

Theorem

$\underline{L}^{f}\subset\underline{L}_{det}\subset\underline{L}\subset\underline{L}^{\lambda}_{det}$ and $\underline{P}^{f}\subset\underline{P}_{det}\subset\underline{P}\subset\underline{P}^{\lambda}_{det}$

Except for this result not much is known when comparing Petri net languages with the deterministic families directly. More information can be obtained by studying their closure properties.

CLOSURE PROPERTIES

In order to understand the properties and limits of Petri net languages as well as to see some of the design methods for new systems from old ones we now consider closure properties of the various Petri net languages, first looking closer to the family of free (terminal) Petri net languages.

When we are given two descriptions of systems and we want to design a system that behaves as either one, then we simply take the union of the two systems and a non-deterministic choice of which to begin with. If we look at Petri net languages, the situation is not always that simple, since the union of free Petri net languages may not be a free language of some other Petri net since we cannot label distinct transitions equally. Thus the following result from [Starke 78] is not surprising:

Lemma

The families $\underline{L}^f,\underline{T}^f,\underline{G}^f,\underline{P}^f$ are not closed with respect to union.

This follows, since the languages {baa} and {aab} are elements of either class but their union is not even contained in $\underline{L}^f$, see [Starke 78].

For the non-free labelled Petri net languages closure with respect to union is trivially true since disjunct union of the systems is always possible.

Also for product the situation for free Petri net languages is not so easy since there are even singletons which are not free terminal Petri net languages, such as the set $L:=\{aba^2ba^3ba^4b\}$, for which P. Starke showed $L \notin \underline{L}^f$. Since we already know from the example before that $\{aba^2ba^3b\}$ is an element of $\underline{L}^f$ we can quickly conclude the following:

Lemma

The family $\underline{L}^f$ of free terminal Petri net languages is not closed with respect to product.

There are only a few operations that can be used to define new free Petri net languages from given ones. Among those that may not be very common is that of taking a quotient from the left or from the right by finite or arbitrary sets.

Definition

For any languages A,B the left (right, resp.) quotient $B^{-1}A$ (AB^{-1}, resp.) of A by B is the set

$B^{-1}A := \{v \mid \exists w \in A, u \in B : w = uv\}$

$AB^{-1} := \{u \mid \exists w \in A, v \in B : w = uv\}$

If $B=\{w\}$ is a singleton, then the quotient is called the derivative of A by w . Often these operations are denoted by $B\backslash A$ for $B^{-1}A$ and A/B for AB^{-1}.

If we want to take the left-derivative of some free Petri net languages by the transition sequence w , then we take the unique marking reached from the initial marking m_o by this sequence as a new initial marking. A similar consideration for the final markings gives the following result:

Lemma

The family $\underline{L}^f$ is closed with respect to left-derivative and right-quotient by finite sets.

Almost all closure properties which are valid for free Petri net languages are at the same time true for all other families of labelled Petri net languages of the same type. The proofs need only minor modifications.

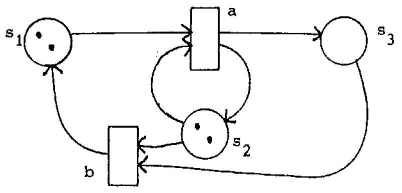
The closure with respect to intersection is easily proved by an effective construction that identifies the transitions of two P/T-systems that are equally labelled or identical.

For proving closure with respect to inverse homomorphisms consider a given P/T-system $N = (S,T,F,W,m)$, a set of final markings $M \subset \mathbb{N}^S$ and a homomorphism $h: X^* \to T^*$. The new P/T-system $N' = (S',X,F',W',m')$ for $h^{-1}(L(N,m,M))$ is constructed as follows: Let $S' := S \cup \{q\}$, where $m'(q) := 1$ and $m''(q)=1$ for each final marking $m'' \in M'$. Now transitions x with $h(x)=\lambda$ get q as its only input and output place, i.e., $^\bullet x = x^\bullet = \{q\}$. If $h(x)=w \in T^+$, then we take x^- as the smallest marking that enables w , the so-called hurdle $H(w) := \min\{m \in \mathbb{N}^{S'} \mid m[w>\}$ and define $x^+ := \Delta(w)+x^-$ so that finally $\Delta(x) = \Delta(w)$, F' and W' are defined accordingly. Also $\forall s \in S$: $m'(s)$ and the final markings differ exactly in the new place q . It is not hard to verify that $L(N',m',M') = h^{-1}(L(N,m,M))$.

Example

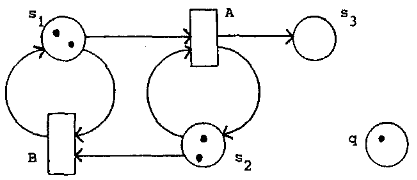
Let N be the P/T-net drawn below together with its initial marking m , $X:=\{A,B\}$, and $h: X^* \to T^*$ given by $h(A):=a$, $h(B):=ab$, then $h^{-1}(T(N,m))=\{BB,ABB,BAB\}$ since $T(N,m)= \{a^2b^2, abab, aba^2b, a^2bab\}$.

N:



The construction, if applied to this P/T-net would give the net together with its initial marking m' as displayed below. However, $T(N',m') = \{AA, ABA, ABB, BB, BAB,$

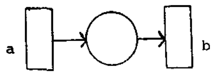
N':



$BAA\} \neq h^{-1}(T(N,m))$. This shows that the construction should be modified if the deadlock languages are to be used. This is sometimes possible by taking appropriate final markings but will not be discussed here in more detail.

It is interesting to see that the family $\underline{L}^f$ of free terminating Petri net languages obtainable from the one-sided Dyck set $D_1 := \{w \in \{a,b\}^* \mid |w|_a = |w|_b$ and $w=uv$ implies $|u|_a \geq |u|_b\}$ also generated as $D_1 = L(N,\underline{0},\underline{0})$ by the following simple P/T-net:

N:



Theorem

$\underline{L}^f$ is the smallest class of languages containing the one-sided Dyck set D_1 and closed with respect to the operations left derivative, right quotient by finite sets, inverse homomorphism, and intersection. $\underline{L}^f$ is not closed with respect to the following operations: union, product, intersection by regular sets, Kleene star, homomorphism.

The free Petri net languages have been seen to form a very weak family of languages which do not even contain all the regular sets. It is therefore appropriate to use the non-free labelled languages from the classes $\underline{L}, \underline{T}, \underline{G}$ or $\underline{P}$. It is easy to see that all but the class $\underline{P}$ contain the regular sets but still we have not seen whether these families are closed with respect to arbitrary or at least nonerasing homomorphisms.

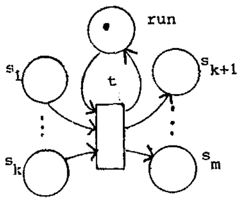
Since we know from the preceeding results and observations that $\underline{L}$ is closed with respect to intersection, intersection by regular sets, inverse homomorphisms, and codings, i.e., length preserving homomorphisms, by definition it follows from the results in formal language theory that this family is also closed with respect to nonerasing homomorphisms, while $\underline{L}^\lambda$ is closed with respect to arbitrary homomorphism. In order to give an idea of how such a proof would be given directly using Petri net constructions we will give the basic steps without all the formal details.

Lemma

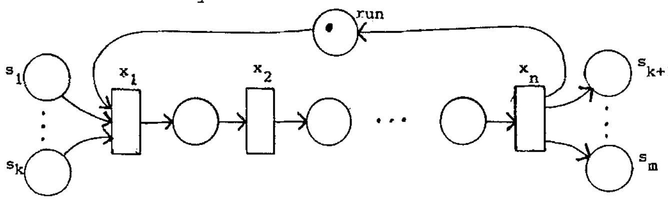
The family $\underline{L}$ (respectively $\underline{L}^\lambda$ is closed with respect to nonerasing (arbitrary, resp.) homomorphism.

For a proof replace the transition t of a P/T-system by $|h(t)|$ new transitions together with the proper codings to yield (h(t) if they occur in sequence. In order to make sure that no other transition in the system can occur inbetween, these transitons must form a sequence and its first transition must disable all other transitions not occurring in this sequence. This is usually done by first introducing a so-called run place which is an input and output place to all transitions in the original P/T-system, and they replace the transition t with $h(t)= x_1,x_2...x_n$ as indicated in the figure below.

Replace



by



Constructions like this obviously change the concurrent behaviour of the underlying nets but they are appropriate for the string language approach. The question as to which operations on languages give raise to closure operators that do not change the various families of Petri net languages is an important one for a comparison with other known families.

For instance, we already know that $\underline{L}$ ($\underline{L}^\lambda$ resp.) is a union and intersection closed trio, i.e., an intersection closed semi-AFL (full semi-AFL, resp.). Moreover, we can identify the parenthesis language D_1 as the single generator from which we can obtain every terminal Petri net language $L \in \underline{L}$ by using the trio operations: inverse homomorphisms, intersection by regular sets, and non-erasing homomorphisms in connection with intersection.

This characterization follows immediately from the following theorem about free (terminal) Petri net languages:

<u>Theorem</u>

Any language $L \in \underline{P}^f$ or $L \in \underline{L}^f$ can be presented in the form $L= L_1 \cap L_2 \cap ... \cap L_k$, where each $L_i, 1 \leq i \leq k$, $k \in \mathbb{N}$ is a deterministic weak one-counter language, i.e. $L_i= g(h^{-1}(D_1) \cap R)$ for some homomorphism h , non-erasing homomorphism g , and a regular set R .

The proof is essentially contained already in [Hack 75] and [Crespi-Reghizzi / Mandrioli 77] but not formulated in this form. For the proof consider those subnets defined by a single place and containing all transitions. The transitions not connected with that place can occur at any time and can be modelled by the finite state control of the deterministic counter automaton which performs the change of its counter according to $\Delta(t)$ for all transitions.

As a consequence of this result we get

Corollary

The complement of a free Petri net language in $\underline{L}^f$ or in $\underline{P}^f$ is a one-counter language but not necessarily a weak one-counter language and not deterministic.

Similar as for deterministic pushdown automata this follows, since the complement of a deterministic weak one-counter language is again one-counter, though in general not deterministic and not weak, see [Harrison 78].

This result does no longer hold if we consider non-free Petri net languages.

From the observation that each place of a given P/T-system behaves like a (weak) counter which cannot be tested for zero except at the end by the final marking, it is not surprising that weak counter languages play the crucial role in those characterizations. In fact, the least intersection closed trio generated by D_1 is known as the family of all languages acceptable by weak (or partially blind) multicounter machines in quasi-real time, [Greibach 78] and thus a machine characterization of (terminal) Petri net languages can easily be given, see also [Jantzen 79 a,b,c].

Since their definition by G. Vidal-Naquet, not many new results were obtained for the class of deterministic Petri net languages. That this family $\underline{L}_{det}$ is not closed with respect to union, product and coding, is widely known among those working in this area but references are not easy to find. The construction used for the intersection of Petri net languages also works for deterministic nets so that $\underline{L}_{det}$ is closed with respect to intersection.

The recent work by E. Pelz [Pelz 86] not only contains proofs for the forementioned but also shows that the complements of deterministic (terminal) Petri net languages are terminal Petri net languages, though not deterministic ones.

Theorem

If $L \subset X^*$ and $L \in \underline{L}_{det}$ or $L \in \underline{P}_{det}$ then $X^* \setminus L \in \underline{L}$.

We will close this section by a table about closure and non-closure properties of the main families of Petri net languages. For the missing proofs see [Hack 75, Peterson 81, Starke 80] and those already mentioned.

operation	$\underline{L}^f$	$\underline{L}$	$\underline{L}_{det}$	$\underline{L}^\lambda$	$\underline{P}^f$	$\underline{P}$	$\underline{P}_{det}$	$\underline{P}^\lambda$
union	-	+	-	+	-	+	-	+
product	-	+	-	+	-	+	-	-
intersection	+	+	+	+	+	+	+	+
intersection by regular sets	-	+	+	+	-	prefix	prefix	prefix
Kleene star	-	-	-	-	-	-	-	-

cont.

operation	$\underline{L}^f$	$\underline{L}$	$\underline{L}_{det}$	$\underline{L}^\lambda$	$\underline{P}^f$	$\underline{P}$	$\underline{P}_{det}$	$\underline{P}^\lambda$
inverse homomorphism	+	+	+	+	+	+	+	+
regular substitution	−	λ-free	λ-free	+	−	prefix	λ-free prefix	prefix
homomorphism	−	−	−	+	−	−	−	+
non-erasing homomorphism	−	+	−	+	−	+	−	+
complement	−	−	−	−	−	−	−	−

COMPARISON WITH OTHER FAMILIES

That $\underline{L}$ is incomparable with the class CF of context-free languages has been shown by Peterson who proved that the language $\{ww^R | w \in \{a,b\}^*\}$ is not in the class $\underline{L}$ by counting the number of different markings that are reachable by transition sequences of length k . Since these are only polynomially many but there are exponentially many different strings of length k , this yields a contradiction. A similar method was used in [Jantzen 81] to show that $\underline{L}$ is not closed with respect to iterated shuffle.

Theorem

(a) Rev:= $\{ww^R | w \in \{a,b\}^* \notin \underline{L}$

(b) $\{ab^n cde^n f | n \geq 1 \quad \notin \underline{L}$

(c) The family $\underline{L}^\lambda$ does not contain all context-free languages, not even the set Rev.

(d) None of the families $\underline{L}$ and $\underline{L}^\lambda$ is closed with respect to Kleene star.

(e) None of the families $\underline{L}$ and $\underline{L}^\lambda$ is closed with respect to complement.

The proofs for (c) and (d) use the decidability of the reachability problem, [Mayr 80, 84], and known results from formal language theory: The family RE of recursively enumerable languages is known to be the smallest intersection closed full AFL containing $L_1 := \{a^n b^n | n \geq 1\}$, and if $\underline{L}$,and consequently then $\underline{L}^\lambda$, would be star-closed, the latter would equal RE . The same would happen if Rev $\in \underline{L}^\lambda$ by results in [Baker/ Book 74].

The non-closure with respect to complement can be shown without recursing to the reachability problem. It is easy to show that the complement $\overline{\text{Rev}}$ of Rev is an element of $\underline{L}$, thus if $\underline{L}$ would be closed under complement, then this contradicts (a). If $\underline{L}^\lambda$ would be this family, then by the proof for (c) $\underline{L}^\lambda$ = RE , but this contradicts the non-closure of RE with respect to complement.

What is clear from the characterization is the inclusion of $\underline{L}$ in the family CS of context sensitive languages, which, by the preceeding result, must be proper. $\underline{L}^\lambda$ was shown to be strictly included in the family of all recursive sets in [Greibach 78] using the decidability of the reachability problem.

The type of proof, showing Rev $\notin \underline{L}$ was used in [Jantzen 79 a,b] to prove that the set BIN:= $\{wa^k | w\in\{0,1\}^*, o\leq k\leq n(w)\}$, where n(w) is the integer represented by w as a binary number, is not an element of $\underline{L}$. Since BIN $\in \underline{L}^\lambda$ we have the following comparison between these classes:

Theorem
$\underline{L} \subsetneq \underline{L}^\lambda$ and $\underline{P} \subsetneq \underline{P}^\lambda$

From this result it becomes apparent that hiding transitions by λ-labels not only is a handy concept for simple formulations, but at the same time it increases the expressibility, at least in terms of Petri net languages.

SPECIFIC PROPERTIES

Apart from the direct comparison of the language families in question, other properties can be used for a classification. Very often languages are considered simple when they are commutatively equivalent to a regular set. A set with this property has a semilinear set of vectors as its image under the Parikh mapping and is sometimes called a slip language (has the semilinear property).

Definition
A set $M \subset \mathbb{N}^k$ is called semilinear if there exists a regular set $R \subset \{X_1,\ldots X_k\}^*$ such that $\psi(R)=M$. A language L as well as a family $\mathcal{L}$ is called slip if $\psi(L)$ is a semilinear vector set (for each $L \in \mathcal{L}$, resp.).

The family CF is such a slip family and there exist slip languages not in $\underline{L}$, see [Jantzen 81], and $\underline{L}$ is not a slip family since the set $\{a^n b^m | 0\leq m\leq 2^n, n\geq 0\}=h(\mathrm{BIN})$ is an element of $\underline{L}$ which is obviously not slip.

The connection between Petri net languages and reachability sets is simple and not very strong. The Parikh image of $L \in \underline{L}^f$ needs only be transformed lineary according to the transition rule:
$[m\rangle = m + C\cdot\psi(L) = m+\Delta(L)$, then, if L is slip, so is $C\cdot\psi(L)$ and consequently $[m\rangle$. The converse is in general not true.

On the other hand, many subclasses of P/T-systems have semilinear reachability sets and actually generate slip languages. These are, among others, the folloiwng classes of P/T-systems the definitions of which are contained in [Araki/Kasami 77], [Grabowski 80], [Jantzen/Valk 79], [Landweber/Robertson 78] and [Mayr 81].

A P/T-system has a semilinear reachability set and a free Petri net language that is slip, if it is
- bounded
- reversible

- persistent
- persistent-reversible
- can be transformed into a pure P/T-system with at most 5 places

This last result is from [Hopcroft/Pansiot 79] and uses the equivalent formulation of pure P/T-systems by vector addition systems.

The full marking class [m] of a P/T-system is always a semilinear set that can be constructed effectively so that it may be presented as the projection of the reachability set of a P/T-system with only bounded or reversal bounded places (see [Jantzen/Valk 79]). However, this is of purely theoretical interest. Let us close this paper by a recent result about Petri net languages and reachability sets of P/T-systems with final markings.

If no final marking is specified, then the following result can be deduced from the covering graph, see [Burkhard 81]:

<ins>Theorem</ins>

Let N= (S,T,F,W,m) be a P/T-system, then its reachability set [m> contains a linear subset which is simultaenously unbounded in exactly the places that are simultaneously unbounded in N .

For languages $L \in \underline{P}^\lambda$ we have a pumping lemma similar to that for regular sets.

<ins>Theorem</ins>

For each $L \in \underline{P}^\lambda$ there exist numbers k,l such that each $w \in L$ with $|w| \geq k$ has a decomposition $w= xyz$ with $1 \leq |y| \leq l$ such that $xy^{i+1}z \in L$ for all $i \in \mathbb{N}$.

This pumping cannot occur in systems where a certain final marking must be reached. However, T.L. Lambert as recently observed the following by using - if not reproving - the decidability of the reachability problem [Lambert 86 b]. Actually, this is already contained but somehow hidden in [Mayr 80].

<ins>Theorem</ins>

Let N= (S,T,F,W,m) be a P/T-system, $M \subset \mathbb{N}^S$ a finite set, then there exist $k_o \in \mathbb{N}, u_i, v_i, w_i, x_i \in T^+$ for $1 \leq i \leq n = |T|$ such that $u_o^k\ v_o\ w_o^k\ x_o^k\ t_1\ u_1^k\ v_1\ w_1^k\ x_1^k t_2 \ \ldots$ $t_n\ u_n^k\ v_n\ w_n^k\ x_n^k \in L(N,m,M)$ for all $k \geq k_o$.

Consequently any infinite language $L \in \underline{L}^\lambda$ has a Parikh image which contains an arithmetic series, which is a special type of semilinear set, and languages like $\{a^n b^m | m=2^n \geq 1\}$, $\{a^n b^m c^k | k=n \cdot m \geq 0\}$ or $\{a^n b^m | m \geq 2^n \geq 1\}$ are not elements of the family $\underline{L}^\lambda$ of arbitrarily labelled Petri nets.

Since Lambert's work is not yet available we can only be vague about the details but it seems that this will be a nice application of solvability of the reachability problem.

Because of the lack of space we could nŏt include the characterizations of $\underline{L}$ and $\underline{G}$ by certain logical first-order formulas. This characterization has as one consequence that this logic might be used as a specification language of a desired behaviour and at the same time a construction of the corresponding labelled Petri net could be constructed.

What we also did not consider here is any enlargement of the classes $\underline{L}$ and $\underline{L}^{\lambda}$ by generalizing the transition rule as done in [Burkhard 81a], [Etzion/Yoeli 81], [Hack 75], [Valk 78], and by others. Usually the extension is that powerful that $\underline{L}^{\lambda}$ becomes the whole class RE . The best known generalizations are probably inhibitor arcs and priority rules.

As shown in [Jantzen 81] the same happens to the family $\underline{L}^{\lambda}$ if powerful closure operations are used, as for instance inverse shuffle or cancellation. Complexity arguments also allow for a classification of the language families but more so for the problems one has to deal with when using Petri nets. The undecidable problems definitely are limitations for the questions that must be solvable effectively.

REFERENCES

The references to this contributio are combined with those of the next one: "Complexity of Place Transition Nets".