
# Algebraic Properties of Petri Net Languages and Codes

**PhD Dissertation**

**Yoshiyuki Kunimochi**

**University of Debrecen**
**Faculty of Informatics**

**Debrecen, 2009**

I developed this dissertation in the frame of the program "Foundations of Informatics" of Doctoral School of Informatics at Debrecen University, Faculty of Informatics to obtain the doctoral (PhD) degree of Debrecen University, Faculty of Informatics.

.....
sign of applicant

This certifies that the applicant Yoshiyuki Kunimochi has done his work by my supervision in the frame of the program "Foundations of Informatics" of the above mentioned Doctoral School. By his work based on independent research the applicant dominantly contributed to the results contained in the dissertation. I recommend the acceptance of the dissertation.

.....
sign of supervisor

# Contents

# Chapter 0

# Foreword

My study have closely related to the field of concurrency and systems since I began to study the graduate research at Shizuoka University. This literature mainly involves the topics of Petri nets and codes, which I studied during working at my institute.

During I was a senior student or a graduate school student at Shizuoka University, I studied the implementation methods of parallel reduction for a pure functional language with lazy evaluation. In those days, such researches which aimed to realize the fifth generation computers were intensively and eagerly promoted by the Japan Ministry of International Trade and Industry. An approach of dataflow model/machine was one method for parallel computation, which I studied in the master course. In the doctoral course, I studied reduction machine model, which is the other approach for parallel computation. Finally I suggested a parallel reduction algorithm for a pure lazy functional language. But I could not implement the algorithm by a real hardware, for lack of a fund and a labor, and went through many hardships.

I run across the Peterson's book[23] when I was a graduate school student. Then though I found that Petri nets had something in common with my research, I was not interested in Petri nets. In those day I studied at extension and application of existential dataflow models, Petri nets were too simple to realize my aim and had poor power as a machine model.

With the passage of time, I studied the theory of automata, formal languages, codes, algebra, computation, etc., and I have recognized more computational power and interest of Petri nets than I expected, for all its primitive action.

I want to express great thanks to my supervisor, Professor Pál Dömösi, who warmly recommended to develop the dissertation. I would like to thank my co-authors of the joint papers which have been the basis of the dissertation, that is, Professor Genjiro Tanaka, Professor Toshimitsu Inomata and Professor Masami Ito. I studied the Petri net structure in the joint work[52],[51] writ-

ten in Chapter 3 with Professor Tanaka, who is my colleague, and Professor Inomata, who was my colleague and now works at Iwate Prefectural University. The properties of Petri net and codes in written in Chapter 4 is the joint work[38] with Professor Ito at Kyoto Sangyo University. Without their help, support and contribution, this dissertation could not be completed.

Fukuroi, April 2009
Yoshiyuki Kunimochi

# Chapter 1

# Introduction

Petri nets are graphical and mathematical modeling tools applicable to many systems. They are promising tools for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic.

Historically speaking, the concept of the Petri net has its origin in Carl Adam Petri's dissertation[41], submitted in 1962 to the faculty of Mathematics and Physics at the Technical University of Darmstadt, West Germany.

However in many applications modeling by itself is of limited practical use if one cannot analyze the modeled system. As means of gaining a better understanding of the Petri net model, the decidability and computational complexity of typical automata theoretic problems concerning Petri nets have been extensively investigated in the past four decades.

A Petri net $(N, \mu_0)$ where $N = (P, T, W)$ is regarded as a directed, weighted, bipartite graph, together with an initial marking $\mu_0$. Each node of the graph is either a place (an element of $P$) or a transition (an element of $T$), and each arc (an element of $(P \times T) \cup (T \times P)$) has a nonnegative integer given by the weight function $W$. The initial marking (state) $\mu_0$ is nonnegative integers accompaning places. A marking is called positive if these numbers are positive. In order to simulate the dynamic behavior of a system, starting from the initial marking, a new marking is derived from a current marking in succesion by applying an enable transition according to the transition (firing) rule. A marking obtained in the derivation is called reachable from the initial marking $\mu_0$. Then the sequence of applied transitions is called a firing sequence. A firing sequence is positive if each marking obtained in the derivation is positive. We denote the set of firing sequences and positive firing sequences by $L(N, \mu_0)$ and $L_+(N, \mu_0)$

Two types of properties can be studied with a Petri net model: those which depend on the initial marking, and those which are independent of the initial marking. The former type of properties is referred to as marking-dependent or behavioral properties whereas the latter type of properties is called structural properties

Reachability is a fundamental basis for studying the behavioral properties. The *reachability problem* for Petri nets is the problem of finding if for a given

marking $\mu$, $\mu$ is reachable from the initial marking. The *single-place zero-reachability problem* is the problem whether for given Petri net $PN$ and a given place $p$ there exists a reachable marking $\mu$ from the initial marking such that $\mu(p) = 0$. These two dicision problems are equivalent[15]. After then, it has been shown that the reachability problem is decidable [27][35] although it takes at least exponential space(and time) to verify in the general case. However, the equality problem [15],[17] is undecidable, i.e., there is no algorithm for determining if $L(N, \mu_0) = L(N', \mu_0')$ for any two Petri nets $(N, \mu_0)$ and $(N', \mu_0')$.

A language over an alphabet $X$ is a subset of the free monoid $X^*$ generated by $X$. A language of our interest is mainly determined by some procedure, computation and derivation, and so on. By applying the concept of a automaton to a Petri net, the Petri net generates a language[16], called a Petri net language, which is at most context-sensitive. Originally the motivations are to check and validate a system by analyzing the language generated by all possible sequences (words) of system actions, and to automatically synthesis a Petri net that accepts only words of a specific language.

There are twelve well-known classes of Petri net languages are defined[23], which are classified by four types (L,G,T,P) of final conditions and by three types (free, $\lambda$-free and with $\lambda$-transition) of labelings of transitions. A Petri net language is composed of firing sequences leading to their respective final conditions, and modified by a labeling of transitions. An L-type is defined by all firing sequences leading to one of final markings like a finite state machine. A G-type is defined by all firing sequences leading to a marking greater than or equal to some final marking. A T-type Petri net language are defined by identifying the set of final states used in the definition of L-type languages with the set of terminal states. A P-type Petri net language is composed of all (reachable) firing sequences and becomes a prefix language.

Recently in some papers Petri net controlled grammars are suggested, which are context-free grammars where the derivation is regulated by a Petri net. The class of languages generated by Petri net controlled grammars is equivalent to the class of languages generated by matrix grammars. That is, it includes the class of context-free languages but is included in the class of context-sensitive languages.

A language $L$ is called a code if it freely generates the submonoid $L^*$ in $X^*$. A prefix code $L$ is a code which no word in $L$ is a proper left factor of any other word in $L$. G.Tanaka defined four types (S, D, C, B) of prefix codes based on Petri nets[47]. We focus on C-type Petri net (CPN, for short) codes, especially CPN codes which is a maximal prefix codes.

The CPN code $\mathbb{C}(N, \mu_0)$ generated by a Petri net $(N, \mu_0)$ is the set of all nonpositive firing sequences in $(N, \mu_0)$ whose proper prefixes are all positive firing sequences instead. That is, $\mathbb{C}(N, \mu_0) = L \setminus LX^+$, where $L = L(N, \mu_0) \setminus L_+(N, \mu_0)$. The notion of maximality of a CPN code is very important in relation to liveness or deadlock. $\mathbb{C}(N, \mu_0)$ is a maximal prefix code or $\mathbb{C}(N, \mu_0) = \emptyset$ if and only if all of transitions are enabled under any marking reachable from $\mu_0$ through a positive firing sequence in $(N, \mu_0)$. This condition is obviously true if $(N, \mu_0)$ is input-ordinary. Conversely we wonder whether the set of all positive firing sequences in a general Petri net $(N, \mu_0)$

with $\mathbb{C}(N, \mu_0)$ being a maximal prefix code is identical with the set of all positive firing sequences in some input-ordinary Petri net $(N_1, \mu_1)$, such that $L_+(N, \mu_0) = L_+(N_1, \mu_1)$.

Chapter 2 plays a role of the introduction for us to the basic notion and of the reference of definitions and notation throughout the literature. These contents is mainly owed to [39], [53]. At first, we introduce the definition of a Petri net and its related concepts and notation. After then the applications, the special subsets and the computational problems of Petri nets are refered. one can skip these references since they are not so deeply related to the remaining chapters. Next, we explain the basic concepts of automata and formal languages and Petri net languages according to [23]. Finally, we introduce four types of prefix codes generated by Petri nets and their fundamental properties[47].

Chapter 3 is spent on Petri net structures representing finite groups, which are treated in [52, 51]. This is a joint work with Professor Genjiro Tanaka and Professor Toshimitsu Inomata. Here the problem of automorphism groups of nets is discussed. We construct a net, called a transformation net, from a transformation semigroup in a similar way that we construct an automaton without outputs from a transformation semigroup. It is well known that for a given group $G$ there exists an automaton such that its automorphism group is isomorphic to $G$. This fact is proved by using the property that the right regular representation of a group $G$ commutes with the left regular representation of $G$ as a permutation group on $G$. An analogous method is applied to prove our main result which states that for a given finite group $G$ there exists a net $N$ such that its automorphism group $\mathbf{Aut}(N)$ is isomorphic to $G$. That is, we construct a transformation net which corresponds to the right regular representation of a given group $G$ and we show that $\mathbf{Aut}(N)$ is isomorphic to $G$ by making some arc-weight of the net in certain conditions.

In Chapter 4 we discuss only about CPN codes introduced in Chapter 2. This is a joint work[38] with Professor Masami Ito. Various properties of finite maximal CPN codes are investigated and two operations $\oplus$(some kind of parallel operation) and $\otimes$(some kind of interruption) are introduced. The property being a maximal CPN code over $X$ is not preserved under concatenation, $\oplus$ and $\lambda$-free homomorphism but is preserved under $\otimes$. In the other half of the chapter, we treat the generative power of CPN codes. There it is shown that there exists a CPN code which is not context-free, but arbitrary CPN code is a context-sensitive language. We will prove that a CPN code is a context-sensitive language in two different ways.

In Chapter 5 we treat the open question raised in Chapter 4. This chapter is completely my own work. A CPN code generated by some input-ordinary Petri net is called an input-ordinary CPN code and it is obviously a maximal CPN code. The question is whether **iCPNC** = **mCPNC** or not, where **iCPNC** and **mCPNC** means the families of maximal CPN codes and input-ordinary CPN codes, respectively. It is easily seen that the latter is a subfamily of the former. But the reverse inclusion is still open in a general Petri net. So we show that the inclusion is true in restricted cases, i.e., the case that the number of places is $\leq 2$, and the case that the number of transitions is equal to 1. The general case still remains open.

# Chapter 2

# Definitions and Notation

This chapter plays roles of the introduction for us to the basic notion and of the reference of definitions and notation throughout the literature. At first, we introduce the definition of a Petri net and its related concepts and notation. After then the applications, the special subsets and the computational problems of Petri nets are referred. One can skip these subsections since they are not so deeply related to the remaining contents. These contents is mainly owed to [39],[53]. Next, we explain the basic concepts of automata and formal languages and Petri net languages according to [23]. Finally, we define four types of prefix codes generated by Petri net and their fundamental properties [47].

## 2.1 Petri net

We introduce the definition of a Petri net and summarize its applications, subclasses, analyzing method, and so on.

### 2.1.1 Definitions and Notation

In this section, we give definitions and fundamental properties related to Petri nets. First of all, a Petri net is viewed as a particular kind of directed graph, together with an initial state $\mu_0$, called the *initial marking*. The underlying graph $N$ of a Petri net is a directed, weighted, bipartite graph consisting of two kinds of nodes, called *places* and *transitions*, where arcs are either from a place to a transition or from a transition to a place.

**DEFINITION 2.1.1 (Petri net)** A *Petri net* $PN$ is a 4-tuple, $PN = (P, T, W, \mu_0)$ where
(1) $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places,
(2) $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions,
(3) $W : E \to \{0, 1, 2, 3, \ldots\}$, i.e.,$W \in {\boldsymbol{N}_0}^E$, is a *weight function*, where $E = (P \times T) \cup (T \times P)$,
(4) $\mu_0 : P \to \{0, 1, 2, 3, \ldots\}$, i.e., $\mu_0 \in {\boldsymbol{N}_0}^P$, is the initial marking,
(5) $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

When a Petri net structure (net, for short) $N = (P, T, W)$ without any specific initial marking is denoted by $N$, a Petri net with a given initial marking $\mu_0$ is denoted by $(N, \mu_0)$. □

**Remark** A Petri net is often given as a 5-tuple $(P, T, F, W, \mu_0)$ adding the set $F$ of flow relations, i.e., arcs with positive weights: $F = \{(p,t) \mid W(p,t) > 0\} \cup \{(t,p) \mid W(t,p) > 0\} \subseteq (P \times T) \cup (T \times P)$[39]. Then, a Petri net structure is also given as 4-tuple $N = (P, T, F, W)$. □

In graphical representation, places are drawn as circles, transitions as bars or boxes. Arcs are labeled with their weights(positive integers), where a $k$-weighted arc can be interpreted as the set of $k$ parallel arcs. Labels for unity weight are usually omitted. A marking (state) assigns a nonnegative integer $k$ to each place. If a marking assigns a nonnegative integer $k$ to place $p$, we say that $p$ is *marked with* $k$ tokens. Pictorially, we put $k$ black dots (tokens) in place $p$. A marking is denoted by $\mu$, an $n$-dimensional row vector, where $n$ is the total number of places. The $p$-th conponent of $\mu$, denoted by $\mu(p)$, is the number of tokens in place $p$.

**EXAMPLE 2.1.1** Figure 2.1 shows a graphical representation of a Petri net. This Petri net $PN = (P, T, W, \mu_0)$ represents a process that a bicycle is assembled from one boby and two wheels. The places are $P = \{\mathbf{body}, \mathbf{wheel}, \mathbf{bicycle}\}$ and the transitions are $T = \{\mathbf{assembly}\}$. Arcs $f_1 = (\mathbf{body}, \mathbf{assembly})$, $f_2 = (\mathbf{wheel}, \mathbf{assembly})$ and $f_3 = (\mathbf{assembly}, \mathbf{bicycle})$ have the weights of 1, 2 and 1, respectively. The other arcs have the weights of 0, and they are not usually drawn in the picture. Note that the weights of $f_1$ and $f_3$ is omitted since they are unity. That is, $W(f_1) = W(f_3) = 1, W(f_2) = 2$, $W(f) = 0$ for each $f \in (P \times T) \cup (T \times P) \setminus \{f_1, f_2, f_3\}$.

The initial marking $\mu_0$ is often denoted by a vector $\mu_0 = (4, 3, 0)$. The place **body** is marked with three tokens. Then we usually put the number of tokens in a place, instead of black dots(tokens). □

In Chapters 4 and 5, we will discuss an input-ordinary Petri net defined in the following definition. The concept of an input-ordinary Petri net is deeply related to the maximality of a Petri net code.

**DEFINITION 2.1.2 (ordinary Petri net)** A Petri net $PN = (P, T, W, \mu_0)$ is called *input-ordinary* (resp., *output-ordinary*) if $W(p,t) \leq 1$ (resp., $W(t,p) \leq 1$) for each $p \in P$ and $t \in T$. A Petri net is called *ordinary* if it is input-ordinary and output-ordinary. □

**DEFINITION 2.1.3 (positive marking)** A marking is *positive* if it is a function from $P$ to $\boldsymbol{N}_0 \setminus \{0\}$. □

The behavior of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behavior of a system, a

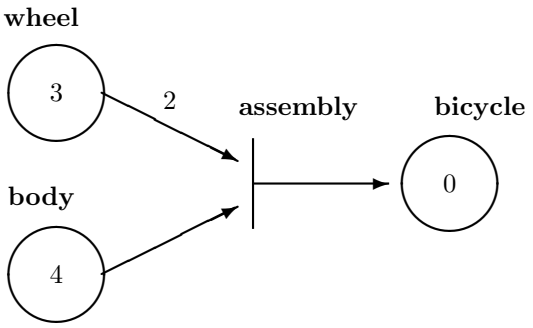



Figure 2.1: Graphical representation of a Petri net

state or marking in a Petri net $PN = (P, T, W, \mu)$ is changed according to the following transition (firing) rule:
(1) A transition $t \in T$ is said to be *enabled* (under the marking $\mu$ or under the Petri net $PN$) if $W(p,t) \leq \mu(p)$ for every place $p \in P$, where $W(p,t)$ is the weight of the arc from $p$ to $t$. Then each input place $p$ of $t$ is marked with at least $W(p,t)$ tokens. An enabled transition may or may not fire (depending on whether or not the event actually takes place).
(2) A *firing* of an enabled transition $t$ removes $W(p,t)$ tokens from each input place $p$ of $t$, and adds $W(t,p)$ tokens to each output place $p$ of $t$. As a consequence of the firing, the current marking $\mu$ is replaced with the following new marking $\mu'$:

$$\mu'(p) = \mu(p) - W(p,\, t) + W(t,\, p) \;\text{ for }\; \forall p \in P. \tag{2.1}$$

For the equation (2.1), we often use the notation $\mu\ [t\rangle\ \mu'$ (or $(N,\, \mu)\ [t\rangle\ (N,\, \mu')$, to emphasize the underlying net).
(3) A sequence $\sigma = t_1 t_2 \ldots t_n$ of transitions is said to be a *firing sequence* of a Petri net $PN = (P, T, W, \mu)$ if $\mu_0 = \mu$, $\mu_n = \mu'$, and $\mu_{i-1}\ [t_i\rangle\ \mu_i$ for each $i\,(1 \leq i \leq n)$. Then we often also use the notation $\mu\ [\sigma\rangle\ \mu'$. In particular, a firing sequence $\sigma$ is said to be *positive* if all $\mu_i\,(1 \leq i \leq n)$ are positive.
(4) A marking $\mu$ is said to be *reachable* from the initial marking $\mu_0$ if there exists a firing sequence $\sigma$ such that $\mu_0\ [\sigma\rangle\ \mu$. Then $\mu$ is said to be reachable from $\mu_0$ through $\sigma$. The set of all possible markings reachable from $\mu_0$ in a Petri net $(N, \mu_0)$ is denoted by $R(N, \mu_0)$ or simply $R(\mu_0)$. The set of all possible firing sequences from $\mu_0$ in a Petri net $(N, \mu_0)$ is denoted by $L(N, \mu_0)$ or simply $L(\mu_0)$. The set of all possible positive firing sequences from $\mu_0$ in $(N, \mu_0)$ is denoted by $L_+(N, \mu_0)$ or simply $L_+(\mu_0)$.

The *transition function* (or *next-state function*) $\delta_{PN}$ of the Petri net $PN = (N, \mu)$ is defined by

$$\begin{array}{ll} \delta_{PN}(\mu_0, \sigma) = \mu' & \text{if } \mu_0 \ [\sigma\rangle \ \mu', \\ \delta_{PN}(\mu_0, \sigma) \text{ is undefined} & \text{if } \mu' \notin R(\mu_0). \end{array}$$

We may denote $\delta_{PN}$ by $\delta$ if no confusion is possible.

**EXAMPLE 2.1.2** Consider the Petri net $PN = (P, T, W, \mu_0)$ shown in Figure 2.1. The transition **assembly** is enabled under the initial marking $\mu_0 = (4, 3, 0)$. Once the transition fires, the marking changes from $\mu_0$ to $\mu_1 = (3, 1, 1)$. Then the transition **assembly** is not enabled under $\mu_1$ because $W(\textbf{wheel}, \textbf{assembly}) = 2 \leq \mu(\textbf{wheel}) = 1$ does not hold. They cannot assemble any more bicycle due to lack of wheels.

So the sequences 1(the empty sequence) and **assembly** are firing sequences of $PN$ but **assembly assembly** isn't a firing sequence. As concerns the next-state function $\delta$ of $PN$, $\delta(\mu_0, 1) = (4, 3, 0)$, $\delta(\mu_0, \textbf{assembly}) = (3, 1, 1)$, $\delta(\mu_0, \textbf{assembly assembly})$ is undefined. □

For ease of expression, the following notation will be used extensively throughout the literature. Let $(P, T, W, \mu)$ be a Petri net, $p \in P$ be a place, $t \in T$ be a transition and $\sigma$ be a transition sequence. We implicitly assume that some orderings $p_1 \leq p_2 \leq \ldots \leq p_n$ and $t_1 \leq t_2 \leq \ldots \leq t_m$ on $P = \{p_1, p_2, \ldots, p_n\}$ and $T = \{t_1, t_2, \ldots, t_m\}$ are established, respectively.

$\#_\sigma(t)$ represents the number of occurrences of $t$ in $\sigma$. For convenience, we sometimes treat $\#_\sigma$ as an $m$-dimensional (row) vector , that is, a $1 \times m$ matrix $(\#_\sigma(t_1), \#_\sigma(t_2), \ldots, \#_\sigma(t_m))$.

$Tr(\sigma) = \{t | t \in T, \#_\sigma(t) > 0\}$, denoting the set of transitions occurring in $\sigma$.

$\Delta(t)$ is the *displacement* of $t$, that is, $n$-dimensional row vector whose $i$-th component is the value of $-W(p_i, t) + W(t, p_i)$. The $i$-th component of $\Delta(t)$ is often denoted by $\Delta(t)(p_i)$. We denote $\Sigma_{i=1}^{k} \Delta(s_i)$ by $\Delta(\sigma)$ where $\sigma = s_1 s_2 \ldots s_k \, (s_i \in T)$. That is, $\Delta(\sigma) = \delta(\mu_0, \sigma) - \delta(\mu_0, 1)$ if $\sigma$ is a firing sequence of the Petri net.

The following $n \times m$ matrix is called the *incidence matrix* of the Petri net.

$$T = ({}^t\Delta(t_1) \, {}^t\Delta(t_2) \, \ldots \, {}^t\Delta(t_m)) = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1m} \\ d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nm} \end{pmatrix}$$

where ${}^t\Delta(t_j)$ is the transpose of $\Delta(t_j)$ and $d_{ij} = \Delta(t_j)(p_i) (1 \leq i \leq n, 1 \leq j \leq m)$.

If $\sigma$ is a firing sequence from the initial marking $\mu_0$ to $\mu$, the following equation holds using the incidence matrix $T$.

$$^t\mu_0 + T^t\#_\sigma = {}^t\mu.$$

We use the following symbols for a pre-set and a post-set of a place $p \in P$ or a transition $t \in T$:

$\bullet t = \{p \in P | W(p,t) > 0\}$ is the set of input places of $t$.

$t \bullet = \{p \in P | W(t,p) > 0\}$ is the set of output places of $t$.

$\bullet p = \{t \in T | W(t,p) > 0\}$ is the set of input transitions of $p$.

$p \bullet = \{t \in T | W(p,t) > 0\}$ is the set of output transitions of $p$.

A transition $t$ (a) without any output place (i.e., $\bullet t \neq \emptyset$ and $t \bullet = \emptyset$ ) (b) with at least one input places and at least one output places (i.e., $\bullet t \neq \emptyset$ and $t \bullet \neq \emptyset$), (c) without any input place(i.e., $\bullet t = \emptyset$ and $t \bullet \neq \emptyset$), or (d) without any input place and any output place ($\bullet t \cup t \bullet = \emptyset$) is called a *sink*, *transform*, *source* or *isolated* transition, respectively.

Note that a source transition is unconditionally enabled, and that the firing of a sink transition consumes tokens, but does not produce any.

Similarly a place $p$ is called (a) *sink*, (b) *transform*, (c) *source* or (d) *isolated* place if $\bullet p \neq \emptyset$ and $p \bullet = \emptyset$, $\bullet p \neq \emptyset$ and $p \bullet \neq \emptyset$, $\bullet p = \emptyset$ and $p \bullet \neq \emptyset$ or $\bullet p \cup p \bullet = \emptyset$, respectively.

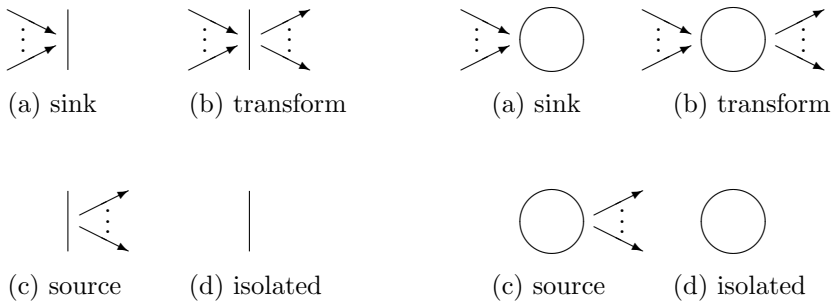



Figure 2.2: Classification of places and transitions

A pair of a place $p$ and a transition $t$ is called a *self-loop* if $p$ is both an input and output place of $t$. A Petri net is said to be *pure* if it has no self-loops.

### 2.1.2 Applications

Here we can see the some kinds of Petri nets used in the application field.

A. Finite capacity net

For modeling many physical systems, it is natural to consider an upper limit to the number of tokens that each place can hold. Such a Petri net is referred to as a *finite capacity net*. A Petri net stated so far can be said to have an infinite capacity. For a finite capacity net $(N, \mu_0)$, every place $p$ has an associated *capacity* $K(p)$, the maximum number of tokens that $p$ can hold at any time. In a finite capacity net the transition rule is modified to be *strict*. That is, a transition is enable if an condition that the number of tokens in every place $p$ cannot exceed its capacity $K(p)$ after firing $t$, in addition to the (weak) transition rule, which we usually use.

This rule with the capacity constraint is called the strict transition rule, whereas the rule without the capacity constraints called the (weak) transition rule. Given a finite capacity net $(N, \mu_0)$, it is possible to apply either the strict transition rule to the given net $(N, \mu_0)$ or, equivalently, the weak transition rule to a transformed net $(N', \mu_0')$, the net obtained from $(N, \mu_0)$ by the *complementary-place transformation*, where it is assumed that $N$ is pure.

**EXAMPLE 2.1.3 (Complementary-place transformation)** Let $(N, \mu_0)$ be the capacity net shown in Figure 2.3 with the capacity $K(p) = 4$. Since the strict transition rule is applied, $a, aba$ are firing sequences but $a^2$ is not. By the complementary-place transformation, adding to this net the place $\bar{p}$, called the *complementary-place* of $p$, and two arcs $(\bar{p}, a)$ and $(b, \bar{p})$ connected to $\bar{p}$, we have the new net $(N', \mu_0')$ shown in Figure 2.4, where $\mu_0'(p) = \mu_0(p) = 1, \mu_0'(\bar{p}) = K(p) - \mu_0(p) = 4 - 1 = 3$, weights of $(a, p)$ and $(\bar{p}, a)$ are identical, and weights of $(p, b)$ and $(b, \bar{p})$ are so. In the net $(N', \mu_0')$ without capacity constraint, $a, aba$ are firing sequences but $a^2$ is not. We can easily see that the firing sequences of these two net are identical.

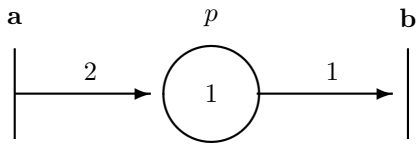



Figure 2.3: Finite capacity net with capacity $K(p) = 4$

EXAMPLE 2.1.3 can be summarized in the following theorem.

**THEOREM 2.1.1** Let $(N, \mu_0)$ be a pure finite-capacity net, where the strict transition rule is to be applied. Let $(N', \mu_0')$ be the net obtained from $(N, \mu_0)$ by the complementary-place transformation, where the weak transition rule is applicable to $(N', \mu_0')$. Then the two nets $(N, \mu_0)$ and $(N', \mu_0')$ are equivalent in the sense that both have the same set of all possible firing sequences.

B. Finite-state machines

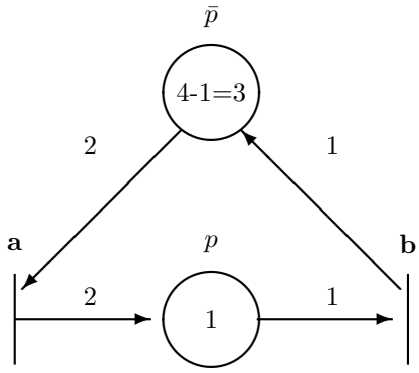



Figure 2.4: Infinite capacity net;**4-bounded,** $L4$**-live=live**

Finite-state machines or their state diagrams can be equivalently represented by a subclass of Petri nets. As an example of a finite-state machine, we consider a task scheduling (management) in an operating system shown in Figure 2.5. On a new task being produced, the task scheduler initially puts it into the ready state. The token in the ready place in the figure means the initial state. By occurrence of an event(for example, time slice interruption), the task transits to the run state and its computation begins. Once it invokes an I/O instruction, it moves to the wait state and remain in the state until the I/O finishes to work. After then it goes to the ready state again.

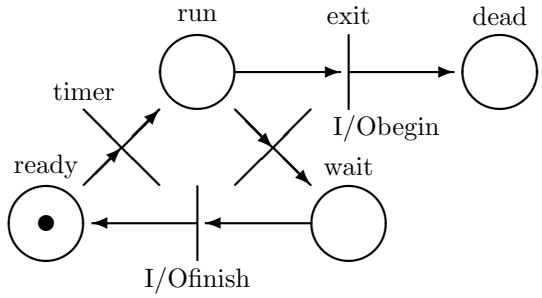



Figure 2.5: State machine for task scheduling; **1-bounded=safe, strictly** $L1$**-live** (∵"timer","I/Obegin" and "I/Ofinish" are $L3$-live while "exit" is $L1$-live)

Note that each transition in this net has exactly one incoming arc and exactly one outgoing arc. The subclass of Petri nets with this property is known as

state machines. Any finite-state machine (or its state diagram) can be modeled with a state machine.

The structure of the place "run", having two (or more) output transitions "exit" and "I/Obegin" shown in Figure 2.5, is referred to as a conflict, decision, or choice, depending on applications. State machines allow the representation of decisions, but not the synchronization of parallel activities.

C. Parallel activities

Parallel activities or concurrency can be easily expressed in terms of Petri nets. For example, in the Petri net shown in Figure 2.6, the parallel or concurrent activities represented by transitions "concentrate"-"bore" and "eat" begin at the firing of transition "TVon", and end with the firing of transition "TVoff". In general, two transitions are said to be *concurrent* if they are causally independent, i.e., one transition may fire before or after or in parallel with the other, as in the case of "concentrate" and "eat", or "bore" and "eat" Figure 2.6.

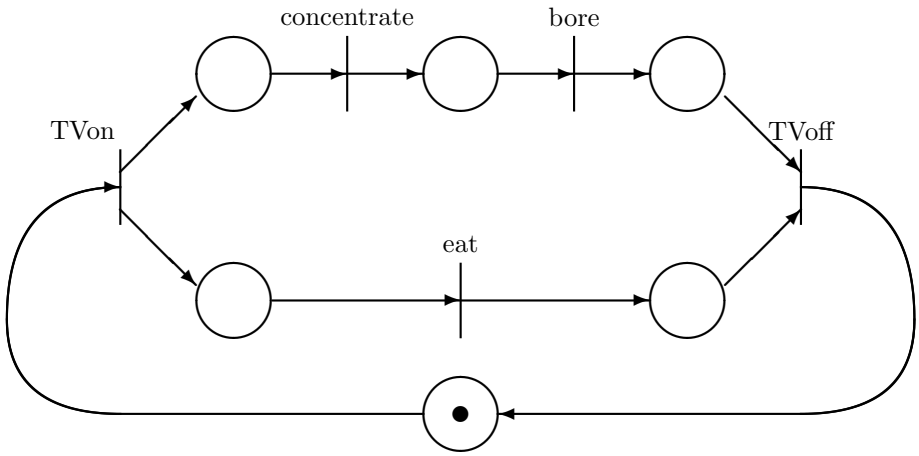



Figure 2.6: Parallel activities; **1-bounded=safe**, $L4$**-live=live**

Note that each place in the net shown in Figure 2.6 has exactly one incoming arc and exactly one outgoing arc. The subclass of Petri nets with this property is known as marked graphs. Marked graphs allow representation of concurrency but not decisions (conflicts).

D. Dataflow Computation

Petri nets can be used to represent not only the flow of control but also the flow of data. The net shown in Figure 2.7 is a Petri net representation of a dataflow computation. A dataflow computer is one in which instructions are enabled for execution by the arrival of their operands, and may be executed

concurrently. In the Petri net representation of a dataflow computation, tokens denote the values of current data as well as the availability of data. ln the net shown in Figure 2.7, the instructions represented by transitions "multiply" and "subtract" can be executed concurrently and deposit the resulting data $a * b$ and $a - b$ into the respective output places.

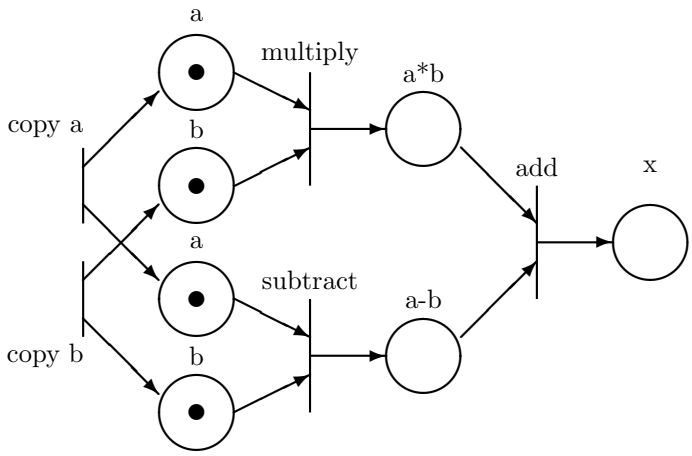



Figure 2.7: Dataflow Computation; **1-bounded=safe**, $L4$**-live=live**

E. Synchronization Control

In a multiprocessor or distributed processing system, resources and information are shared among several processors(,processes or tasks). This sharing must be controlled or synchronized to insure the correct operation of the overall system. Petri nets have been used to model a variety of synchronization mechanisms, including the mutual exclusion(semaphore), readers-writers problem.

The Petri net shown in Figure 2.8 represents a semaphore, which is a well-known programming mechanism devised by Edsger Dijkstra for restricting accesses to shared resources. If at most $k$ tasks can access a shared resource, the semaphore for the resource is called a counter semaphore. In this case, $k$ tokens are initially stored in the place "semaphore". The P-instruction removes just one token from the place "semaphore" if there are some tokens in it and puts it into the place "a" or "b". Otherwise the P-instruction wait until a tokens appears in the place "semaphore". The V-instruction removes just one token from the place "a" or "b" and adds it into the place "semaphore". The P- and V-instructions must be atomic instructions.

F. Formal Languages

When the transitions in a Petri net are labeled with a set of not necessarily distinct symbols, a firing sequence generates a string of symbols. The set of

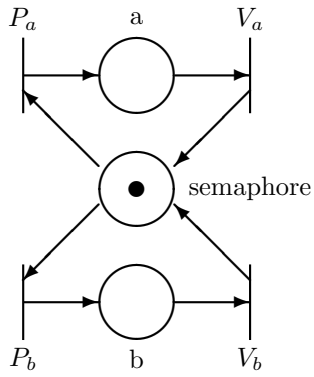



Figure 2.8: Semaphore; **1-bounded=safe,** $L4$**-live=live**

strings generated by all possible firing sequences defines formal language called a *Petri net language* (See Section 2.4). In the Petri net shown in Figure 2.9, transitions are labeled with 1(the empty word), $a$ and $b$. The initial marking gives only one token to the place "start". The final markings leaves only one token at the place "end". The set of string generated by all possible firing sequences from the initial marking to the final marking are

$$L = \{1, ab, a^2b^2, \ldots, a^ib^i, \ldots\}.$$

$L$ is a context-free Petri net language. Since every finite-state machine can be modeled by a Petri net, every regular language is a Petri net language. It has been shown that all Petri net languages are context-sensitive languages [23].

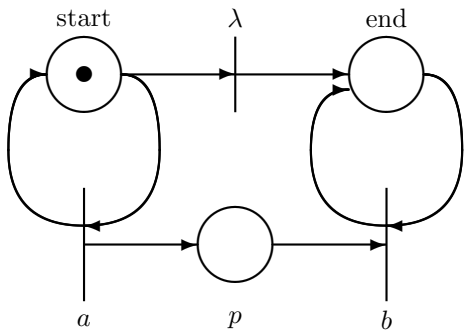



Figure 2.9: Petri net generating $\{a^ib^i \mid i \geq 0\}$; **not bounded,** $L1$**-live** ($\because$ $a$ is $L3$-live, $b$ is $L2$-live, $\lambda$ is $L1$-live)

### 2.1.3 Subclasses of Petri Nets

Suppose that any Petri net considered in this section is ordinary, and moreover that it has no isolated places and tarnsitions. For a subset $S$ of places, $\bullet S$ denotes the union of all $\bullet p$ with $p \in S$.

**DEFINITION 2.1.4 (state machine(SM))**
An ordinary Petri net $PN = (P, X, W, \mu_0)$ is called a *state machine*(SM) if each transition $t \in T$ has exactly one input place and exactly one output place, i.e.,

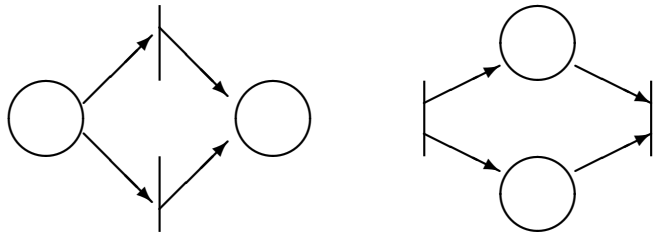
$$|\bullet\, t| = |t\, \bullet| = 1 \quad \text{for all} \quad t \in T.$$

**DEFINITION 2.1.5 (marked graph(MG))**
An ordinary Petri net $PN = (P, X, W, \mu_0)$ is called a *marked graph*(MG) if each place $p \in P$ has exactly one input transition and exactly one output transition, i.e.,

$$|\bullet\, p| = |p\, \bullet| = 1 \quad \text{for all} \quad p \in P.$$

□

Figure 2.10 (a) is an SM but not a MG, while (b) is a MG but not an SM.



(a) SM,$\overline{\text{MG}}$ (b) MG,$\overline{\text{SM}}$

Figure 2.10: Examples of SM and MG

**DEFINITION 2.1.6 (free-choice net(FC))**
An ordinary Petri net $PN = (P, X, W, \mu_0)$ is called a *free-choice net*(FC) if every arc from a place is either a unique outgoing arc or a unique incoming arc to a transition, i.e.,

$$|p\, \bullet| \leq 1 \quad \text{or} \quad \bullet\,(p \bullet) = \{p\} \quad \text{for all} \quad p \in P. \tag{2.2}$$

□

Note that the FC condition (2.2) is equivalent to the following condition (2.3):

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Longrightarrow |p_1 \bullet| = |p_2 \bullet| = 1 \quad \text{for all distinct} \quad p_1, p_2 \in P. \tag{2.3}$$

Assume that the condition (2.2) holds. Suppose that $t \in p_1 \bullet \cap p_2 \bullet \neq \emptyset$ with $p_1 \neq p_2$. Then $\bullet(p_1 \bullet) = \{p_1\}$ leads a contradiction that $\{p_1, p_2\} \subseteq \{p_1\}$. Since $PN$ is an FC, $|p_1 \bullet| = 1$. Similarly we also have $|p_2 \bullet| = 1$. On the other hand, assume that the condition (2.3) holds. Let $|p \bullet| > 1$. $p, q \in \bullet(p \bullet)$ implies that $p \bullet \cap q \bullet \neq \emptyset$, and that we have $p = q$ by the conditions (2.3) and $|p \bullet| > 1$. Hence $\bullet(p \bullet) = \{p\}$.

In other words, if two transitions share some input places, then they share all their input places.

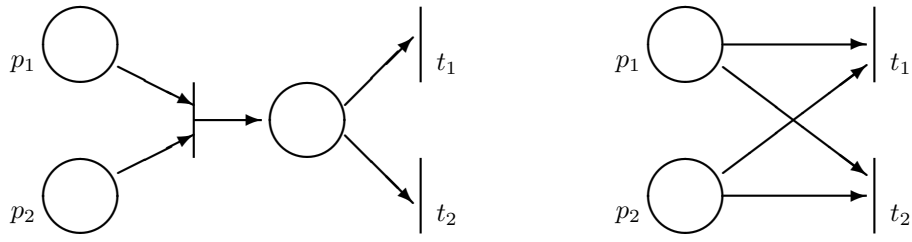
**DEFINITION 2.1.7 (extended free-choice net(EFC))**
An ordinal Petri net $PN = (P, X, W, \mu_0)$ is called a *extended free-choice net*(EFC) if

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Longrightarrow p_1 \bullet = p_2 \bullet \quad \text{for all} \quad p_1, p_2 \in P.$$

□

We show that an FC $PN = (P, X, W, \mu_0)$ is an EFC. Since $p_1 = p_2$ obviously implies the conclusion, then $p_1 \neq p_2$. If $p_1 \bullet \cap p_2 \bullet \neq \emptyset$, $|p_1 \bullet| = |p_2 \bullet| = 1$ by the condition (2.3) since $PN$ is an FC. Then we have $p_1 \bullet = p_2 \bullet = \{t\}$

Figure 2.11 (a) is an FC net but neither an SM nor an MG, while (b) is an EFC but not an FC.




(a) FC,$\overline{\text{SM}}$,$\overline{\text{MG}}$ (b) EFC,$\overline{\text{FC}}$

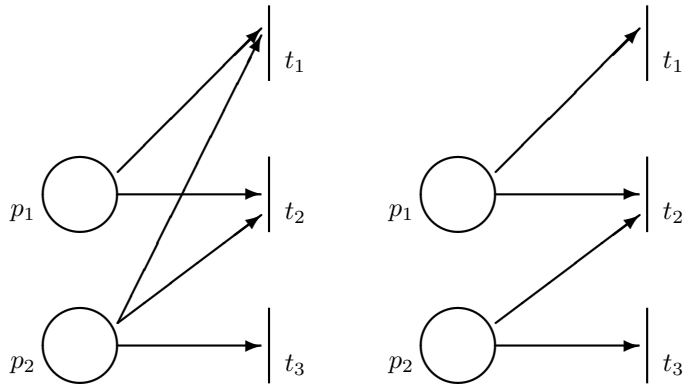
Figure 2.11: Examples of FC and EFC

**DEFINITION 2.1.8 (asymmetric choice net(AC))**
An ordinary Petri net $PN = (P, X, W, \mu_0)$ is called an *asymmetric choice net*(AC) if

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \implies p_1 \bullet \subseteq p_2 \bullet \quad \text{or} \quad p_2 \bullet \subseteq p_1 \bullet \quad \text{for all} \quad p_1, p_2 \in P. \tag{2.4}$$

□

Figure 2.12 (a) is an AC net but not an EFC because $\emptyset \neq p_1 \bullet \cap p_2 \bullet = \{t_1, t_2\} \cap \{t_1, t_2, t_3\}$ but $p_1 \bullet \neq p_2 \bullet$. While (b) is a PN but not an AC because $\emptyset \neq \{t_2\} = p_1 \bullet \cap p_2 \bullet$ but $t_1 \in p_1 \bullet - p_2 \bullet$ and $t_3 \in p_2 \bullet - p_1 \bullet$.




(a) AC,$\overline{\text{EFC}}$ (b) PN,$\overline{\text{AC}}$

Figure 2.12: Examples of AC and PN

Figure 2.13 shows the hierarchy among classes of state machines(SMs), marked graphs(MGs), free-choice nets(FCs), extended free-choice nets(EFCs), asymmetric choice nets(ACs), and ordinary Petri nets(PNs). SMs→ FCs means that the set of SMs is included in the set of FCs. As we have seen, these inclusions are all strict.

### 2.1.4 Behavioral Properties

After modeling systems with Petri nets, an obvious question is " What can we do with the models?" A major strength of Petri nets is their support for analysis of many properties and problems associated with concurrent systems. Two types of properties can be studied with a Petri net model: those which depend on the initial marking, and those which are independent of the initial marking. The former type of properties is referred to as marking-dependent or behavioral properties whereas the latter type of properties is called structural properties. In this section, we discuss only basic behavioral properties and their analysis problems.

A. Reachability

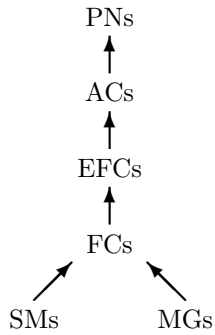



Figure 2.13: Hierarchy among subclasses of Petri nets

Reachability is a fundamental basis for studying the dynamic properties of any system. Recall that the firing of an enabled transition will change the token distribution (marking) in a net according to the transition rule described in Section 2.1.1. The set of all possible markings reachable from $\mu_0$ in a Petri net $(N, \mu_0)$ is denoted by $R(N, \mu_0)$ or simply $R(\mu_0)$. The set of all possible firing sequences from $\mu_0$ in a Petri net $(N, \mu_0)$ is denoted by $L(N, \mu_0)$ or simply $L(\mu_0)$.

Now, the *reachability problem* for Petri nets is the problem of finding if $\mu \in R(\mu_0)$ for a given marking $\mu$. ln some applications, one may be interested in the markings of a subset of places and not care about the rest of places in a Petri net. This leads to a submarking reachability problem which is the problem of finding if $\mu' \in R(\mu_0)$, where $\mu'$ is any marking whose restriction to a given subset of places agrees with that of a given marking $\mu$. It has been shown that the reachability problem is decidable [27][35] although it takes at least exponential space(and time) to verify in the general case. However, the equality problem [16],[15],[17] is undecidable, i.e., there is no algorithm for determining if $L(N, \mu_0) = L(N', \mu_0')$ for any two Petri nets $(N, \mu_0)$ and $(N', \mu_0')$.

B. Boundedness

A Petri net $(N, \mu_0)$ is said to be *k-bounded* if the number of tokens in each place does not exceed a finite number $k$ for any marking reachable from $\mu_0$, i.e., $\mu(p) \leq k$ for every place $p$ and every marking $\mu \in R(\mu_0)$. A Petri net $(N, \mu_0)$ is said to be *bounded* if there exists some integer $k$ such that $(N, \mu_0)$ is $k$-bounded. A Petri net $(N', \mu_0')$ is said to be *safe* if it is 1-bounded. For example, the nets shown in Figures 2.4, 2.5, 2.6, 2.7 and 2.8 are all bounded; in particular, the net in Figure 2.4 is 4-bounded and the rest of the nets are safe. On the other hand the net shown in Figure 2.9 is not bounded since infinitely many tokens appear in the place $p$ after $a$ fires as many times. Places in a Petri net are often used to represent buffers and registers for storing intermediate data. By verifying that the net is bounded or safe, it is guaranteed that there will be no overflows in the buffers or registers, no matter what firing sequence

is taken.

Now, the *boundedness problem* for Petri nets is the problem deciding whether if a given Petri net $(N, \mu_0)$ is bounded. This problem was first considered by Karp and Miller[25], where it was shown to be decidable by using the technique of coverability graph analysis.

C. Liveness

The concept of liveness is closely related to the complete absence of deadlocks in operating systems. A Petri net $(N, \mu_0)$ is said to be *live* (or equivalently $\mu_0$ is said to be *live marking*, no matter what marking has been reached from $\mu_0$, it is possible to ultimately fire ary transition of the net by progressing through some further firing sequence. This means that a live Petri net guarantees deadlock-free operation, no matter what firing sequence is chosen.

Liveness is an ideal property for many systems. However, it is impractical and too costly to verify this strong property for some systems such as the operating system of a large computer. Thus, we relax the liveness condition and define different levels of liveness as follows [6],[29]. A transition $t$ in a Petri net $(N, \mu_0)$ is said to be:

(0) *dead(L0-live)* if $t$ can never be fired in any firing sequence in $L(\mu_0)$.

(1) *L1-live(potentially firable)* if $t$ can be fired at least once in some firing sequence in $L(\mu_0)$.

(2) *L2-live* if, given any positive integer $k$, $t$ can be fired at least $k$ times in some firing sequence in $L(\mu_0)$.

(3) *L3-live* if $t$ appears infinitely, often in some firing sequence in $L(\mu_0)$.

(4) *L4-live* or *live* if $t$ is $L1$-live for every marking $\mu$ in $R(\mu_0)$.

A Petri net $(N, \mu_0)$ is said to be $Lk$-live if every transition in the net is $Lk$-live, $k = 0, 1, 2, 3, 4$. $Lk$-liveness is the strongest and corresponds to the liveness defined earlier. It is easy to see the following implications:

$$L4\text{-liveness} \Rightarrow L3\text{-liveness} \Rightarrow L2\text{-liveness} \Rightarrow L1\text{-liveness},$$

where $\Rightarrow$ means "implies." We say that a transition is strictly $Lk$-live if it is $Lk$-live but not $L(k+1)$-live, $k = 1, 2, 3$.

Examples of live Petri nets are shown in Figures 2.6, 2.7 and 2.8. On the other hand, the Petri nets shown in Figures 2.5 and 2.9 are not live but strictly $L1$-live. In the Petri net shown in Figure 2.14, $t_0$, $t_1$, $t_2$, $t_3$ and $t_4$ are $L0$-, $L1$-,$L2$-,$L3$- and $L4$-live, respectively.

D. Reversibility and Home State

A Petri net $(N, \mu_0)$ is said to be *reversible* if, for each marking $\mu \in R(\mu_0)$, $\mu_0$ is reachable from $\mu$. Thus, in a reversible net one can always get back to the initial marking or state. In many applications it is not necessary to get back to the initial state as long as one can get back to some (home) state. Therefore, we refax the reversibility condition and define a home state. A marking $\mu'$ is said to be *home state* if, for each marking $\mu$ in $R(\mu_0)$, $\mu'$ is reachable from $\mu$.

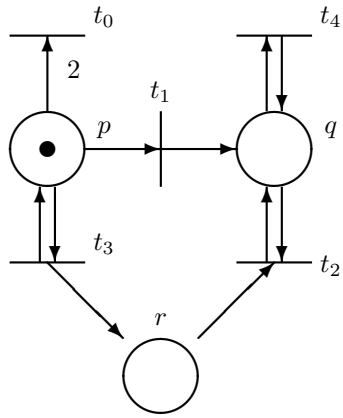



Figure 2.14: Various kinds of livenesses

E. Coverability

A marking $\mu$ in a Petri net $(N, \mu_0)$, is said to be *coverable* if there exists a marking $\mu'$ in $R(\mu_0)$ such that $\mu'(p) \geq \mu(p)$ for each $p$ in the net. The covering problem is the problem deciding whether a given Petri net is coverable or not.

Coverability is closely related to liveness (potential firability). Let $\mu$ be the minimum marking needed to enable a transition $t$. Then $t$ is dead (not live) if and only if $\mu$ is not coverable. That is, $t$ is $L1$-live if and only if $\mu$ is coverable. The complexity analysis of the covering problem is seen for details in [7].

F. Persistence

A Petri net $(N, \mu_0)$ is said to be *persistent* if, for any two enabled transitions, the firing of one transition will not disable the other. A transition in a persistent net, once it is enabled, will stay enabled until it fires. The notion of persistence is useful in the context of parallel program schemata [25] and speed-independent asynchronous circuits [9], [36]. Persistency is closely related to conflict-free nets [31], and a safe persistent net can be transformed into a marked graph by duplicating some transitions and places [42]. Note that all marked graphs are persistent, but not all persistent nets are marked graphs.

G. Synchronic Distance

The notion of synchronic distance is a fundamental concept introduced by C.A. Petri[4]. It is a metric closely related to a degree of mutual dependence between two events in a condition/event system. We define the *synchronic distance* between two transitions $t_1$ and $t_2$ in a Petri net $(N, \mu_0)$ by

$$d(t_1, t_2) = \max|\bar{\sigma}(t_1) - \bar{\sigma}(t_2)| \tag{2.5}$$

where $\sigma$ is a firing sequence starting at any marking $\mu$ in $R(\mu_0)$ and $\bar{\sigma}(t_i)$ is the number of times that transition $t_i$, $i = 1, 2$ fires in $\sigma$. For example, in the net shown in Figure 2.5 $d(\text{ready}, \text{I/Obegin}) = 1$, $d(\text{ready}, \text{exit}) = \infty$, etc. In the net shown in Figure 2.6 transitions "concentrate" and "eat" represent two parallel events, and $d(\text{concentrate}, \text{eat}) = 2$ because after firing "eat" there is a firing sequence $\sigma = \text{concentrate eat TVoff TVoff concentrate}$ in which $\bar{\sigma}(\text{concentrate}) = 2$ and $\bar{\sigma}(\text{eat}) = 0$.

The synchronic distance given by (2.5) represents a well-defined metric for condition/event net [50] and marked graphs. However, there are some difficulties when it is applied to more general class of Petri nets [49].

H. Fairness

Many different notions of fairness have been proposed in the literature on Petri nets. We present here two basic fairness concepts: bounded-fairness and unconditional (global) fairness. Two transitions $t_1$ and $t_2$ are said to be in a *bounded-fair* (or *B-fair*) relation if the maximum number of times that either one can fire while the other is not firing is bounded. A Petri net $(N, \mu_0)$ is are said to be a *B-fair net* if every pair of transitions in the net are in a B-fair relation. A firing sequence $\sigma$ is said to be *unconditionally fair* (or *globally fair*) if it is finite or every transition in the net appears infinitely often in $\sigma$. A Petri net $(N, \mu_0)$ is are said to be an *unconditional (global) fair net* if every firing sequence $\sigma$ from $\mu$ in $R(\mu_0)$ is unconditionally (globally) fair.

### 2.1.5 Analysis Method

In this section, we summarize various techniques useful for analyzing Petri net properties. Our focus is on algebraic techniques, structural analysis and state-space analysis. Other techniques such as simulation and synthesis/reduction are not discussed here. Structural analysis is mainly designed for reasoning about properties of Petri nets that are independent of the initial markings. State-space analysis, on the other hand, allows us to infer properties of Petri nets that are sensitive to the initial markings.

A. Algebraic Techniques

In the framework of using algebraic techniques for reasoning about Petri nets, solving a Petri net problem is reduced to finding a solution for an algebraic (in)equation associated with the Petri net. Due to the nature of this technique, the method is in general efficient (in most cases, polynomial in the size of the Petri net). Unfortunately, this technique generally provides only necessary or sufficient information for either inferring desired properties or ruling out dangerous conditions.

Figure 2.15 highlights the idea behind the technique based on state equations with respect to a Petri net $(P, T, W, \mu_0)$. Assume that the set $P$ of places is given some ordering and a marking $\mu : P \to \boldsymbol{N}_0$ is regarded as a vector under this ordering. For example the initial marking $\mu_0$ is written like $\mu_0 = (2, 4)$.

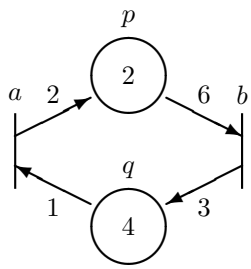



Figure 2.15: State Equation

The first (resp., second) component is the number of tokens in place $p$ (resp., $q$). If $\mu$ is reachable from the initial marking $\mu_0 = (2, 4)$ through transition sequence $\sigma$, then $\#_\sigma = (x, y)$ corresponds to an integer solution to the state equation:

$$\begin{pmatrix} 2 \\ 4 \end{pmatrix} + T \begin{pmatrix} x \\ y \end{pmatrix} = {}^t\mu, \quad \text{where} \quad T = \begin{pmatrix} 2 & -6 \\ -1 & 3 \end{pmatrix}, \tag{2.6}$$

$T$ is the incidence matrix of the Petri net, $\#_\sigma$ represents $m$-dimensional vector whose components are the numbers of occurrences of the transitions in $\sigma$, ${}^t\mu$ is the transpose of an $m$-dimensional vector $\mu$.

Suppose that $\mu = \mu_0 = (2, 4)$. Then $\#_\sigma = (3n, n)$, where $n$ is an integer, is the solution of (2.6). If $\sigma$ is a firing sequence of the Petri net, then $a$ and $b$ must occur $3n$ and $n$ times respectively. But the converse is not necessary true. Indeed $a^2ba, a^3b, \ldots$ are firing sequences of the Petri net but $ba^3, aba^2, \ldots$ are not.

The technique relies on relating the Petri net reachability analysis to integer linear programming, which is a well-established formalism. Unfortunately, a direct application of the state equation technique to general Petri net problems is normally not feasible, as the existence of a solution to a state equation is necessary but not sufficient for witnessing reachability. There are, however, various subclasses of Petri nets for which an extended state equation is sufficient and necessary to capture reachability of the underlying Petri net.

B. Place Invariant

A *place invariant*(P-invariant, for short) of Petri net $(P, T, W, \mu_0)$ with the transition function $\delta$ is a mapping $Inv_P : P \to \boldsymbol{Z}$ (i.e., assigning weights to places) such that for all $\mu, \mu'$ and $t \in T$, if $\delta(\mu, t) = \mu'$, then $\Sigma_{p \in P} Inv_P(p)\mu(p) = \Sigma_{p \in P} Inv_P(p)\mu'(p) \neq 0$. In other words, the firing of any transition does not change the weighted sum of tokens in the Petri net. The Petri net shown in Figure 2.16 (a) has a P-invariant $X$ which is a solution of the following equation (2.7).

$$XT = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 4 & -1 \\ -3 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}, \tag{2.7}$$

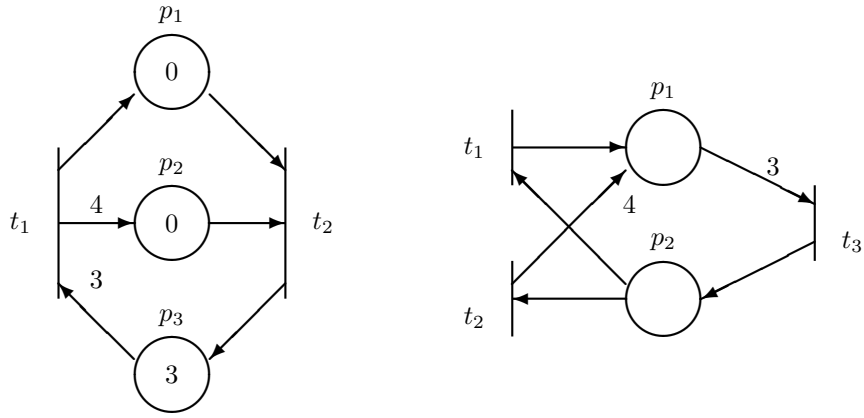
where $T$ is the incidence matrix of the Petri net. The set of solutions $X$ of the equation (2.7) is $\{t(1,2,3)|t \in \boldsymbol{Z}\}$.

Suppose $\mu$ is a reachable marking (from the initial marking $\mu_0$) through a firing sequence $\sigma$. Clearly ${}^t\mu_0 + T^t\#_\sigma = {}^t\mu$. (Here the column vectors ${}^t\#$, ${}^t\mu_0$ and ${}^t\mu$ are transposes of ${}^t\#$, $\mu_0$ and $\mu$.) Let $X$ be a P-invariant. Then

$$X^t\mu = X({}^t\mu_0 + T\,{}^t\#_\sigma) = X^t\mu_0 + XT\,{}^t\#_\sigma = X^t\mu_0.$$

Recall that the Petri net shown in Figure 2.8 has P-invariant $t(-1,1,1)$. For every reachable marking $\mu$, we have $\mu(a) + \mu_0(\text{semaphore}) + \mu(b) = \mu_0(a) + \mu_0(\text{semaphore}) + \mu_0(b)$, meaning that the total number of tokens in $a$, semaphore and $b$ remain unchanged during the course of the Petri net computation. Hence, if the Petri net starts from the initial marking $(0,1,0)$, then the property of mutual exclusion for places $a$ and $b$ can be asserted as $\mu(a) + \mu(b) = \mu_0(a) + \mu_0(b) \leq 1$ for every reachable marking $\mu$.

It is easy to see that if there exists a P-invariant $X$ with $X(p) > 0$ for all $p \in P$, then the Petri net is guaranteed to be structurally boundedness. Hence, place invariants can be used for reasoning about structural boundedness.




(a) P-invariant (b) T-invariant

Figure 2.16: Petri net with P-invariant or T-invariant

C. Transition Invariant

A *transition invariant*(T-invariant, for short) of Petri net $(P, T, W, \mu_0)$ with the transition function $\delta$ is a mapping $Inv_T : P \to \boldsymbol{N}_0$ (i.e., assigning nonnegative weights to transitions) such that $\Sigma_{t \in T} Inv_T(t)\Delta(t) = 0$, where $\Delta(t)$ is the

displacement of $t$. In other words, firing each transition the number of times specified in the T-invariant brings the Petri net back to the same marking. Consider the Petri net shown in Figure 2.16 (b) where $x(1, 2, 3)$ is a T-variant for arbitrary $x \in \boldsymbol{N}_0$. Like P-variants, any linear combination of T-invariants is T-invariants. It is easy to see that T-invariants correspond to the solutions of the following equation:

$$TY = \begin{pmatrix} 1 & 4 & -3 \\ -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tag{2.8}$$

where $T$ is the incidence matrix of the Petri net and $Y$ is a column vector representing a T-variant. The existence of a T-invariant is a necessary condition for a bounded Petri net to be live.

To see this, suppose $PN = (P, T, W, \mu_0)$ is live and bounded. Clearly due to $PN$ being live, there exists an infinite firing sequence $\sigma_1\sigma_2 \ldots \sigma_i \ldots$ such that $\delta(\mu_i, \sigma_i) = \mu_{i+1}$ and $Tr(\sigma_i) = T$ for each $i \in \boldsymbol{N}_0$ where $\delta$ is the next-state function of $PN$. Since $PN$ is also bounded, there exist $0 \leq j < h$ such that $\mu_j = \mu_h$. Hence, $\#_{\sigma_{j+1} \ldots \sigma_h}$ constitutes a T-variant.

D. Structural Analysis

Given a $PN = (P, T, W, \mu_0)$, a subset of places $S \subseteq P$ is called *trap* (resp., *siphon*) if $\bullet S \subseteq S \bullet$ (resp., $S \bullet \subseteq \bullet S$). (Here $\bullet S$ and $S \bullet$ denote the sets of input and output transitions of $S$, respectively.) Intuitively, a trap $S$ represents a set of places in which every transition consuming a token from $S$ must also deposit a token back into $S$. In contrast, if a transition is going to deposit a token to a place in a siphon $S$, the transition must also remove a token from $S$.

Suppose $S$ is a siphon in a live Petri net without isolated places, then $S$ must be marked in the initial marking $\mu_0$ with $\mu_0(p) > 0$ for some $p \in S$. Otherwise, none of the transitions in $S \bullet$ is enabled – violating the assumption that the Petri net is live.

The concept of a siphon plays an important role in the liveness analysis for the class of free-choice Petri nets.(See DEFINITION 2.1.8.) Known as Commoner's Theorem [30], a free-choice Petri net is live iff every nonempty siphon contains an initially marked trap.

E. Reachability Graph Analysis

The so-called *reachability graph analysis* is perhaps the simplest and the most straightforward approach for analyzing the behavior of a Petri net. As its name suggests, such a technique relies on exhaustively generating all the reachable markings from a given initial marking in hope of deducing Petri net properties by examining the structure of the reachability graph. Figure 2.17 shows a reachability graph associated with the Petri net shown in Figure 2.15

In spite of its simplicity, the applicability of the technique of reachability

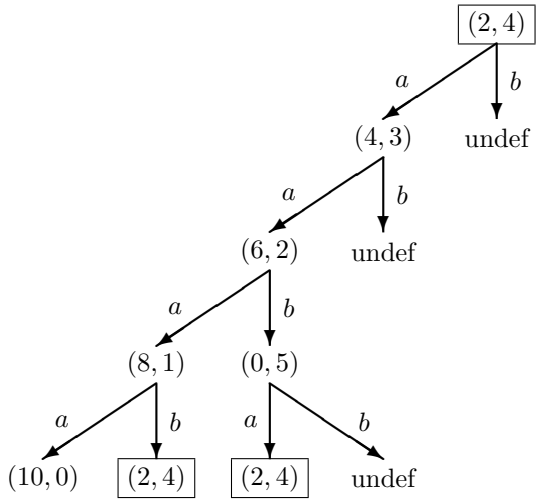



Figure 2.17: Reachability graph of the Petri net shown in Figure 2.15

graph analysis is rather limited; it can only be applied to bounded (finite) Petri nets with small reachability sets. Even for bounded Petri nets which exhibit finite reachability graphs, the technique is expensive in the sense that it suffers from the explosion phenomenon as the sizes of reachability sets grow beyond any primitive recursive function even for bounded Petri nets in the worst case.

F. Coverability Graph Analysis

*Coverability graph analysis* offers an alternative to the technique of reachability graph analysis by abstracting out certain details to make a graph finite. To understand the intuition behind coverability graphs, consider the Petri net shown in Figure 2.18 (a). Consider the path $(1,0,0)$ $[a\rangle$ $(0,1,0)$ $[b\rangle$ $(1,0,1)$ along third coordinate gains an extra token in the end. Clearly the third coordinate can be made arbitrarily large by repeating $ab$ for a sufficient number of times, as

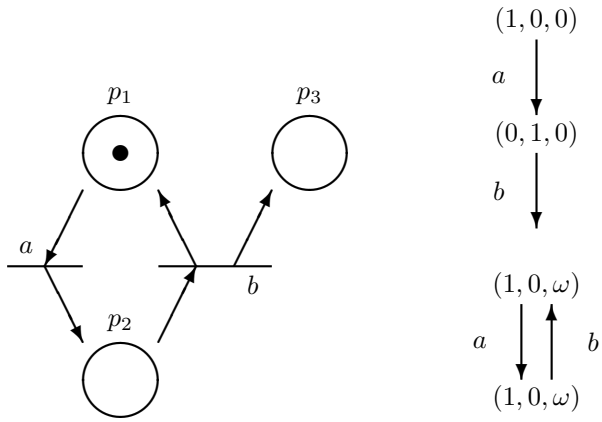
$$(1,0,0)\ [ab\rangle\ (1,0,1)\ [ab\rangle\ (1,0,2)\ [ab\rangle\ \ldots\ [ab\rangle\ (1,0,n)\ [ab\rangle\ \ldots$$

for arbitrary $n$. In order to capture the notion of a place being unbounded, we short-circuit the above infinite sequence of computation as

$$(1,0,0)\ [a\rangle\ (1,0,1)\ [b\rangle\ (1,0,\omega)$$

where $\omega$ is a symbol denoting something being arbitrary large. One can regard $\omega$ as "infinity" having the property that $\omega > n$ for any integer $n$, $\omega + n = \omega + n = \omega$, $\omega - n = \omega$ and $\omega \geq \omega$. A coverability graph relates each node to a

general marking ($\in (\boldsymbol{N}_0 \cup \{\omega\})$) of the original Petri net. The corresponding coverability graph of the Petri net in Figure 2.18 (a) is depicted in Figure 2.18 (b).




(a) Petri net (b) Its coverability graph

Figure 2.18: Petri net and its coverability graph

G. VAS and VRS

There are well-known computational models equivalent to Petri nets. *Vector addition systems(VAS)* were introduced by Karp and Miller[25], and were later shown by Hack[14] to be equivalent to Petri nets. An $n$-dimensional VAS is a pair $G = (x, W)$, where $x \in {\boldsymbol{N}_0}^n$ is called the *start point*(or *start vector*) and $W$ is a finite set of vectors(called *addition vectors*) in $\boldsymbol{Z}^n$. The *reachability set* of the VAS $G$ is the set $R(G) = \{z | \text{ for some } j, z = x + v_1 + \ldots + v_j$, where, for all $1 \leq i \leq j, v_i \in W, x + v_1 + \ldots + v_i \geq 0\}$.

An $n$-dimensional *Vector addition system with states (VASS)* is a VAS $(x, W)$ together with a finite set $T$ of transitions of the form $p \to (q, v)$, where $p$ and $q$ are states and $v \in W$. The computation is performed as follows. Transition $p \to (q, v)$ can be applied at point $y$ in state $p$ yields the point $y + v$ in state $q$, provided that $y + v \geq 0$. The VASS is specified by $G = (x, W, T, p_0)$, where $p_0$ is the starting point.

**EXAMPLE 2.1.4 (VAS)** For the Petri net shown in Figure 2.18 (a), the corresponding VAS $(x, W)$ where $x = (1, 0, 0)$ and $W = \{(-1, 1, 0), (1, -1, 1)\}$.

A $k \times m$ *vector replacement system* (VRS) [26] is a triple $(w_0, U, W)$, where $w_0 \in {\boldsymbol{N}_0}^k$(*starting vector*), $U \in {\boldsymbol{N}_0}^{k \times m}$ (*check matrix*) $W \in \boldsymbol{Z}^{k \times m}$ (*addition matrix*) such that, for any $i, j$ with $1 \leq i \leq m$ and $1 \leq j \leq k$, we have $U_i(j) + W_i(j) \geq 0$. Here $U_i$(resp., $W_i$) is the $i$-th column vector of $U$(resp.,

$W$). A vector $W_i \in W$ is said to be *enabled* in a vector $x \in {\boldsymbol{N}_0}^k$ iff $x \geq U_i$. Then adding $W_i$ to $x$ yeilds $x + W_i \in {\boldsymbol{N}_0}^k$. For a VRS $G = (w_0, U, W)$, $R(G)$ denotes the set of the vectors that can be reached from $w_0$ by iteratively adding vectors from $W$ enabled in the vector computed so far.

For a given $n$-dimensional VASS $G$, we can effectively construct an $(n+3)$-dimensional VAS $G'$ that simulates $G$ [19]. A vector replacement system (VRS) is obviously equivalent to a Petri net.

## 2.2 Algebraic Properties

This section gives us a mathematical and fundamental preparation for languages and codes. In Subsection 2.2.2 we illustrate the regular representation of a group related to the automorphism group of a Petri net structure, which we treat in Chapter 3.

### 2.2.1 Semigroups, Monoids and Groups

A *groupoid* $(S, \mu)$ is defined as a nonempty set $S$ on which a binary operation $\mu$ —by which we mean a map $S \times S \to S$— is defined. We say that $(S, \mu)$ is a *semigroup* if the operation $\mu$ is *associative*, that is to say, if for all $x, y$ and $z$ in $S$,

$$\mu(\mu(x, y), z) = \mu(x, \mu(y, z)) \tag{2.9}$$

(Here, and throughout the book, we write mapping symbols on the left.) This notation is rather cumbersome, and we shall follow the usual algebraic practice of writing the binary operation as multiplication. Thus the image of a pair $(x, y) \in S \times S$ will be written $x \cdot y$ or (more usually) $xy$, and Equation (2.9) takes the simple form:

$$(xy)z = x(yz),$$

i.e., the familiar associative law of elementary algebra. Expressions such as $xyz$ and $x_1 x_2 \cdots x_n$, where $x, y, z, x_1, x_2, \cdots, x_n$ are elements of $S$, then have unambiguous meanings, and we can use the notation $x^n$ ($n \in \boldsymbol{N}$) to mean the product of $n$ elements each equal to $x$. The cardinal number $|S|$ will be called the *order* of $S$.

Where the semigroup is written multiplicatively and where the nature of the multiplication is clear from the context, we shall write simply $S$ rather than $(S, \cdot)$.

If a semigroup $S$ has the property that for all $x, y$ in $S$,

$$xy = yx,$$

then we shall say that $S$ is a *commutative* (or *abelian*) semigroup. If a semigroup $S$ contains an element 1 with the property that, for all $x$ in $S$,

$$x1 = 1x = x,$$

then we say that 1 is an *identity element* (or just an *identity*) of $S$, and that $S$ is a *semigroup with identity* or (more usually) *monoid*. $S$ possesses at most one identity element, since $1'$ also has the property that $x1' = 1'x = x$ for all $x$ in $S$, then they are identical:

$$\begin{aligned} 1' &= 11' && \text{(because 1 is an identity)} \\ &= 1 && \text{(because 1' is an identity).} \end{aligned}$$

An element $s \in S$ is said to be a *unit* if there exist $x, y \in S$ such that $sx = ys = 1$. Then we have $x = y$ because $x = 1 \cdot x = (ys) \cdot x = y \cdot (sx) = y \cdot 1 = y$.

If $A$ and $B$ are subsets of a semigroup $S$, then we write $AB$ to mean $\{ab \mid a \in A, b \in B\}$. It is easy to verify that, for all subsets $A, B, C$ of $S$,

$$(AB)C = A(BC);$$

hence once again notations such as $ABC$ and $A_1A_2 \dots A_n$ are meaningful. The usual hazard, namely that $A^2$ means $\{a_1a_2 \mid a_1, a_2 \in A\}$ rather than $\{a^2 \mid a \in A\}$, should be noted. When dealing with singleton sets we shall use the notational simplifications that are customary in algebra, writing (for example) $Ab$ rather than $A\{b\}$.

If a semigroup $S$ has the property that

$$(\forall a \in S) \quad aS = S \text{ and } Sa = S, \tag{2.10}$$

we call it a *group*. This uncommon definition (2.10) of a group is equivalent to the more usual definition of a group as a semigroup for which

$$\left.\begin{array}{ll} (\exists e \in S)(\forall a \in S) & ea = a, \\ (\forall a \in S)(\exists a^{-1} \in S) & a^{-1}a = e. \end{array}\right\} \tag{2.11}$$

$e$ is called the unity of $S$ and $a^{-1}$ the inverse of $a$. The definition (2.10) is given first because it is the one that seems to occur most often in semigroup theory. It is equivalent to

$$(\forall a, b \in S)(\exists x, y \in S) \quad ax = b \text{ and } ya = b. \tag{2.12}$$

Here we show that (2.11) is equivalent to (2.12). Suppose the condition (2.11) holds. Then for arbitrary $a, b \in S$, setting $x = a^{-1}b$, $y = ba^{-1}$, we have (2.12). Conversely suppose the condition (2.12) holds. Let $a \in S$ be a particular element. Then there exist $e_l, e_r, x, y \in S$ such that $ae_r = a$, $e_la = a$, $ax = e_r$ and $ya = e_l$. Since $e_l = ya = yae_r = e_le_r = e_lax = ax = e_r$ holds, set $e = e_r = e_l$. Let $b \in S$ be an arbitrary element. Since $b \in S$ is written as $b = az = z'a$(for some $z, z' \in S$). $eb = eaz = az = b = z'a = z'ae = be$. This means that $e$ is the unity of $S$. By (2.12), $bx = yb = e$ holds for some $x, y \in S$. Then $x = ex = ybx = ye = y$ implies that $x = y$ is the inverse of $b$.

A map $\phi : S \to T$, where $(S, \cdot)$ and $(T, \cdot)$ are semigroups, is called a *morphism* (or *homomorphism*) if, for all $x, y$ in $S$,

$$\phi(xy) = \phi(x)\phi(y).$$

If $(S, \cdot, 1_S)$ and $(T, \cdot, 1_T)$ are monoids, with identity elements $1_S, 1_T$ respectively, then $\phi$ will be called a morphism only if we have the additional property

$$\phi(1_S) = 1_T.$$

There is a possibility of confusion here, and if there seems any doubt we shall distinguish between a *semigroup morphism* and a *monoid morphism*.

In either case above we refer to $S$ as the *domain* of $\phi$ and $T$ as the *codomain*. The *image* (or *range*) of $\phi$ is defined as $\{\phi(s) \mid s \in S\}$. If $\phi$ is one-to-one we shall call it a *monomorphism*; This definition is equivalent to the 'categorical' definition of a monomorphism as a left cancellative morphism, that is, $\phi : S \to T$ is a monomorphism if, for all semigroup $U$ and for all morphisms $\alpha, \beta : U \to S$,

$$\phi \circ \alpha = \phi \circ \beta \Rightarrow \alpha = \beta$$

where $\circ$ is the composition of mappings. If $\phi$ is onto we call it an *epimorphism* and can interpret left-right dually. The morphism $\phi : S \to T$ is called an *isomorphism* if there exists a morphism $\phi^{-1} : T \to S$ such that both $\phi^{-1} \circ \phi$ and $\phi \circ \phi^{-1}$ are the identity maps of $S$ and $T$, respectively. It is easy to show that a morphism $\phi : S \to T$ is an isomorphism if and only if it is bijective. If there exists an isomorphism $\phi : S \to T$ we say that $S$ and $T$ are *isomorphic*, and write $S \simeq T$.

A morphism $\phi$ from $S$ to itself is called an *endmorphism* of $S$, and if it is one-to-one and onto it is called an *automorphism*.

Just as groups arise most naturally as groups of permutations of a set, so semigroups arise from more general mappings of a set into itself. The analogue of the *symmetric group* $(\mathfrak{S}_X, \circ)$ of all permutations of a set $X$ is the *full transformation semigroup* $(\mathfrak{T}_X, \circ)$ consisting of all maps from $X$ into $X$. The operation in both cases is composition of maps, sometimes written $\circ$, but just written multiplicatively: if $\alpha$ and $\beta$ are maps from $X$ into $X$, then

$$(\alpha \circ \beta)(x) = \alpha(\beta(x)) \quad (x \in X).$$

It is clear that $\mathfrak{S}_X$, consisting of all bijections from $X$ to $X$, is a subgroup of $\mathfrak{T}_X$. Simple combinatorial arguments show that, if $|X| = n$, then

$$|\mathfrak{S}_X| = n!, \quad |\mathfrak{T}_X| = n^n$$

If a semigroup $S$ is a subsemigroup of $\mathfrak{T}_X$ for some $X$, then we say that $S$ is a *semigroup of maps*, or a *transformation semigroup*. A morphism $\phi$ from a semigroup $S$ into some $\mathfrak{T}_X$ is called a *representation* of $S$ (by map), and a representation $\phi$ is called *faithful* if it is one-to-one. In such a case the image $\phi(S)$ of $\phi$ is a transformation semigroup isomorphic to $S$.

### 2.2.2 Regular Representation

In Chapter 3, we show that for a given finite group $G$ there exists a net $N$ such that its automorphism group $\mathbf{Aut}(N)$ is isomorphic to $G$ by using the following *Cayley's theorem* for group.

**THEOREM 2.2.1** *(Cayley)*Every group is isomorphic to a group of permutations.

(Proof) Let $G$ be a group and $g$ be an arbitrary element of $G$.

At first we show that the mapping $\phi_g : G \to G,\ x \mapsto xg$ is a permutation of $G$, that is $\phi_g$ is a bijection. If $\phi_g(x) = \phi_g(y)$, then $xg = yg$. So, multiplying by $g^{-1}$ on the right, we find $x = y$ and $\phi_g$ is an injection. Next suppose $x \in G$; then $\phi_g(xg^{-1}) = (xg^{-1})g = x$ and $\phi_g$ is a surjection.

Secondly we prove that the mapping $\rho : G \to \mathfrak{S}_G,\ x \mapsto \phi_g$ is a homomorphism and an injection.

$$\rho(gh)(x) = x(gh) = (xg)h = \rho(h)(\rho(g)x) = \rho(h) \circ \rho(g)(x).$$

$\rho(g) = \rho(h)$ implies that the images of the identity $e$ of $G$ by $\rho(g)$ and $\rho(h)$ are identical. Therefore $g = h$, i.e., $\rho$ is a injection.

Finally $\rho(G)$ includes the identity mapping $\phi_e$. And an arbitrary $\phi_g \in \rho(G)$ has its inverse $\phi_{g^{-1}}$, where $g^{-1}$ is the inverse of $g$ in $G$. Hence $\rho(G)$ is a group of permutations. □

The map $\rho : G \to \mathfrak{S}_G,\ x \mapsto \phi_g$ in the proof above is a faithful representation of $G$ because $\rho$ is one-to-one. $\rho$ is called the *right regular representation* of $G$.

We slightly touch on the generalization of the right regular representation of a group. Let $G$ be a group and $H$ be its subgroup. A subset of the form $Ha$ (resp., $aH$) ($a \in G$) is a right (resp., left) coset of $H$ in $G$. The right cosets of $H$ in $G$ are the equivalence classes under the equivalence relation $\sim$ on $G$ given by $a \sim b$ if and only if $ab^{-1} \in H$. Similar statements are also true for left cosets. The set of all the distinct right cosets of $H$ in $G$ is denoted by $H\backslash G$. Then $\rho : G \to \mathfrak{S}_{(H\backslash G)},\ g \mapsto \rho(g)$, where $\rho(g) : Hx \mapsto Hxg$, is a representation of $G$. Especially, If $H = 1$, then $\rho$ is faithful (one-to-one) and just becomes the right regular representation of $G$.

The following theorem for semigroup, closely analogous to Cayley's theorem for group. That is, every semigroup is isomorphic to a transformation semigroup.

**THEOREM 2.2.2** *(Cayley)*If $S$ is a semigroup and $X = S^1$ then there is a faithful representation $\rho : S \to \mathfrak{T}_X$.

### 2.2.3 Subsemigroups and Submonoids

A nonempty subset $T$ of a semigroup $(S, \cdot)$ is a *subsemigroup* of $S$ if it is closed under the operation $\cdot$ of $S$. A subsemigroup $N$ of a monoid is called a *submonoid* of $M$ if $N$ contains the identity 1 of $M$.

**EXAMPLE 2.2.1** Let $X = \{1, 2, 3\}$ and $M$ be the full transformation semigroup $\mathfrak{T}_X$ with the identity $e$. $M$ becomes a monoid. The mappings $f$, $g$ from $X$ to $X$ are defined by

$$\begin{array}{lll} f: & 1 \mapsto 1, & 2 \mapsto 1, \quad 3 \mapsto 1, \\ g: & 1 \mapsto 1, & 2 \mapsto 2, \quad 3 \mapsto 2. \end{array}$$

Since $f^2 = f, fg = gf = f, g^2 = g$ hold, $N = \{f, g\}$ is a subsemigroup of $M$. Though $g$ behaves like the identity in $N$, note that $N$ is not a submonoid of $M$ because $g \neq e$. □

Let $A$ be a subset of a semigroup $S$ and $B$ be a subset of a monoid $M$ with the identity 1.

$$A^+ = A \cup A^2 \cup \ldots \cup A^n \cup \ldots,$$

where $A^n = \{a_1 a_2 \ldots a_n | a_i \in A, 1 \leq i \leq n\}$, is obviously a subsemigroup of $S$, called the subsemigroup of $S$ *generated* by $A$.

$$B^* = 1 \cup B^+ = 1 \cup B \cup B^2 \cup \ldots \cup B^n \cup \ldots,$$

is a submonoid of $M$, called the monoid of $M$ generated by $B$. Then $A$ and $B$ are said to be a *generating set* of $S$ and $M$, respectively.

Now, let $S$ be a semigroup without identity. $S$ has at least one generating set($S$ generates itself). A generating set $A$ of $S$ is *irreducible* if and only if no proper subset of $A$ generates $S$. An element $s$ of $S$ is *indecomposable* if and only if $s$ cannot be represented in the form $s = xy$ with $x, y \in S$. It follows that if $s$ is indecomposable, then $s$ must be a member of every generating set of $S$. If the set of all indecomposable elements generates $S$, then it is the unique irreducible generating set of $S$.

**EXAMPLE 2.2.2** Let $S = \{x \in \boldsymbol{R} \,|\, x > 0\}$ be a semigroup with additive operator $+$. We show that $S$ has no irreducible generating set. Suppose that $A$ is an irreducible generating set of $S$ and $r_1, r_2 \in A\,(r_1 < r_2)$. $r = r_2 - r_1 \in A$ implies that $r_2 = r + r_1$, $A \setminus \{r_2\}$ is also generating set of $S$. On the other hand $r \notin A$, as $r$ is written by a product $r = a_1 + a_2 + \ldots + a_n$ of element $a_i \in A\,(1 \leq i \leq n)$. Therefore $r_2 = r_1 + a_1 + a_2 + \ldots + a_n$, $A \setminus \{r_2\}$ is also generating set of $S$. This contradicts the definition of irreducibility. Hence $S$ has no irreducible generating set. □

A semigroup (monoid) with length is a semigroup (monoid) $S$ in which every $s \in S$ is assigned a unique nonnegative integer $\lg(s)$, called the length of $s$, which obeys the laws:

$$\begin{aligned} &\lg(xy) = \lg(x) + \lg(y) \quad \text{for all } x, y \in S, \\ &\lg(x) = 0 \iff x \text{ is the identity element of } S. \end{aligned}$$

**PROPOSITION 2.2.1** [3] Every semigroup $S$ with length has a unique irreducible generating set $\hat{A} = S \setminus S^2$, contained in every other generating set and consisting of all indecomposable element of $S$, i.e., $S = (S \setminus S^2)^+ = \hat{A}^2$. □

**PROPOSITION 2.2.2** [3] Every monoid $M$ with length has a unique irreducible generating set $\hat{A}$ contained every other generating set and consisting of all the element of $M$ that have no proper decomposition. Also $\hat{A} = (M \setminus 1) \setminus (M \setminus 1)^2$. □

### 2.2.4 Free Monoids

A nonempty subset $B$ of a monoid $M$ is a *base*[*] if, whenever

$$u_1u_2\dots u_m = v_1v_2\dots v_n$$

with $u_1, u_2, \dots, u_m, v_1, v_2, \dots, v_n \in B$, then $m = n$ and $u_i = v_i$ for $i = 1, 2, \dots, m = n$. Clearly, the identity 1 may not belong to a base, nor any product of elements of a base equals 1. If a subset $B$ of a monoid $M$ is a base and $C \subseteq M$ is a subset of $M$ such that $B^* = C^*$, then it follows from the definition of a base $B \subseteq C$. If $C$ is also a base, then $B = C$.

A monoid $M$ is *free* if and only if there exists a base $B$ such that $M = B^*$. In view of the above remark, $B$ is called the *base* of $M$. An element $x$ of a free monoid $M$ is uniquely represented as a product of elements $b_i (1 \leq i \leq n)$ of $B$, $x = b_1b_2\dots b_n$. Then if the length of $x$ is defined by $\lg(x) = n$, it satisfies the laws of length. A monoid $M$ has a unique irreducible generating set $\hat{A} = (M \setminus 1) \setminus (M \setminus 1)^2$ by PROPOSITION 2.2.2. Since $B$ is the base of $M$ and $\hat{A}$ is irreducible, we have $B = \hat{A} = (M \setminus 1) \setminus (M \setminus 1)^2$.

Let $M$ be a monoid. For any subsets $A$ and $B$ of $M$, we define

$$A^{-1}B = \{x \in M \mid Ax \cap B \neq \emptyset\},$$
$$AB^{-1} = \{x \in M \mid A \cap xB \neq \emptyset\}.$$

The following characterization of a free monoid is due to Schützenberger.

**THEOREM 2.2.3** [45] Let $S$ be a submonoid of a free monoid $M$. Then $S$ is free if and only if $S^{-1}S \cap SS^{-1} \subseteq S$.

## 2.3 Languages and Codes

The formal language theory began with the study of N. Chomsky to investigate the study of linguistics. He supposed generative grammars, which generates languages with rewriting systems. Another approach to generate languages is the recognition by automata.

We explain terms and notations related to the formal language theory in Section 2.3.1, and codes in Section 2.3.2

### 2.3.1 Formal Languages

We call a (finite or infinite) set of letters (or symbols) an *alphabet*. Through the literature we use $X$ as an alphabet only if we don't specify specially.

A finite sequence of letters in $X$ is called a *word* (or *string*) over $X$. The *empty word*, that is, the word contains no letter, will be denoted by 1. The number of letters occurring in a word $x$ is called the length of $x$ and denoted by $|x|$. In particular, $|1| = 0$ and $|x| = 1 \iff x \in X$.

---

[*]This condition is equivalent to the following implication:

$$(\forall u, v \in B)(uB^* \cap vB^* \neq \emptyset \Rightarrow u = v).$$

The set of all words over $X$ attended with a binary associative operation $\cdot$ defined by *juxtaposition*, sometimes called *concatenation*;

$$(a_1a_2\dots a_m)\cdot(b_1b_2\dots b_n)=a_1a_2\dots a_mb_1b_2\dots b_n,$$

forms the semigroup with the identity 1, that is, is the monoid generated by $X$:

$$X^*=1\cup X\cup X^2\cup\ldots\cup X^n\cdots.$$

The base of $X^*$ is obviously the alphabet $X$. Therefore $X^*$ is free, called the *free monoid generated by* $X$. $X^+=X^*\setminus 1$ is a semigroup, called the *free semigroup generated* by $X$.

**EXAMPLE 2.3.1** Let $X=\{a,\,b\}$. Note that $\{a,b,ab\}^*=X^*$ is the free monoid generated by $X$ but $\{a,b,ab\}$ is not the base $X$ of $X^*$. The submonoid $S_1=\{a,ab,ba\}^*$ of $X^*$ is not free because $aba$ has two distinct factorizations $aba=a\cdot ba=ab\cdot a$. The submonoid $S_2=\{a^2,ab,ba,b^2\}^*$ of $X^*$ is free and $\{a^2,ab,ba,b^2\}$ is the base of $S_2$. □

A subset $L$ of $X^*$ is called a *language* over $X$. A Language of our interest is mainly produced by a mechanical way, that is, computation and derivation, and so on. Here we explain generative grammars which produce languages.

Chomsky grammars

A *phase-structure* (or type 0) grammar is a quadruple $G=(N,X,S,P)$, where $N,X$ are disjoint alphabets, $S\in N$, $P\subseteq V^*NV^*\times V^*$, for $V=N\cup X$. The elements of $N$ are called *nonterminal* symbols, those of $X$ are called *terminal* symbols, $S$ is the *start symbol* or the *axiom*, and $P$ is the set of *production rules*; $(u,v)\in P$ is written in the form $u\to v$.

For $x,y\in V^*$ we write $x\Rightarrow_G y$ iff $x=x_1ux_2, y=x_1vx_2$, for some $x_1,x_2\in V^*$ and $u\to v\in P$. If $G$ is understood, we write $x\Rightarrow y$. The reflective and transitive closure of the relation $\Rightarrow$ is denoted by $\Rightarrow^*$. The *language generated* by $G$ is $\{x\in X^*\mid S\Rightarrow^* x\}$, denoted by $\mathcal{L}(G)$.

A phase-structure grammar $G=(N,X,S,P)$ is called:

*monotonous* (or *length-increasing*) if for all $u\to v\in P$ we have $|u|\le|v|$.

*context-sensitive*(or type 1) if each $u\to v\in P$ has $u=u_1Au_2, v=u_1xu_2$ for $u_1,u_2\in V^*, A\in N, x\in V^+$.

*context-free* (or type 2) if each production $u\to v\in P$ has $u\in N$.

*linear* if each rule $u\to v\in P$ has $u\in N$ and $v\in X^*\cup X^*NX^*$.

*right-linear* if each rule $u\to v\in P$ has $u\in N$ and $v\in X^*\cup X^*N$.

*left-linear* (or type 3) if each rule $u\to v\in P$ has $u\in N$ and $v\in X^*\cup NX^*$.

*regular* if each rule $u\to v\in P$ has $u\in N$ and $v\in X\cup XN\cup\{\lambda\}$.

In monotonous and context-sensitive grammars a production $S \to \lambda$ is allowed, providing $S$ does not appear in the right-hand members of rules in $P$.

The family of languages generated by monotonous grammars is equal with the family of languages generated by context-sensitive grammars. The family of languages generated by right- or left-linear grammars are equal with the family of languages generated by regular grammars.

A language generated by phase-structure (resp., context-sensitive, context-free, regular) grammar is called a phase-structure (resp., context-sensitive, context-free, regular) language and can be accepted by a Turing machine (resp., a linear-bounded Turing machine, a pushdown automaton, a finite automaton).

We denote by $RE, CS, CF, LIN, REG$ the family of languages generated by phase-structure, context-sensitive, context-free, linear and regular grammars, respectively.

The well-known Chomsky hierarchy, that is, the following strict inclusion, holds: $REG \subset LIN \subset CF \subset CS \subset RE$.

### 2.3.2 Codes

Let $X^*$ be the free monoid generated by an alphabet $X$. A *code* is the base of a free submonoid $M$ in $X^*$, and conversely it freely generates the submonoid $M$ in $X^*$. It is formally defined as follows:

A nonempty language $C$ is a code if for any two integers $n$, $m \geq 1$ and any words $u_1, u_2, \cdots, u_n, v_1, v_2, \cdots, v_m \in C$,

$$u_1 u_2 \cdots u_n = v_1 v_2 \cdots v_m$$

implies

$$n = m \quad \text{and} \quad u_i = v_i \text{ for } i = 1, \cdots, n.$$

If for two words $w$, $u \in X^*$ there exists some word $v \in X^*$ with $w = uv$ (resp., $w = vu$), then $u$ is called a *prefix* (resp., *suffix*) or a *left factor* (resp., *right factor*) of $w$, and denoted by $u \leq_p w$ (resp., $u \leq_s w$). A prefix $u$ (resp., a suffix $u$) of $w$ is called *proper* if $u \neq w$, and denoted by $u <_p w$ (resp., $u <_s w$). A word $u$ is a *subword* of a word $w$ if there exist words $v_1$ and $v_2$ (possibly empty) such that $w = v_1 u v_2$.

A language $L$ becomes a code, called a *prefix code*, if $u$, $uv \in L$ implies $v = 1$ (equivalently $L \cap LX^+ = \emptyset$). A *suffix code* is defined left-right dually. A prefix and suffix code is called a *bifix code*. A language $L$ becomes a bifix code, called a *infix code* if $u$, $v_1 u v_2 \in L$ implies $v_1 = v_2 = 1$. A nonempty subset $U$ of $X^n = \{w \,|\, |w| = n\}$ for some positive integer $n$ is a infix code, called a *uniform code*. Especially the uniform code $U = X^n$ is called a *full uniform code*. These codes have the following strict implication:

$$\text{full uniform} \Rightarrow \text{uniform} \Rightarrow \text{infix} \Rightarrow \text{bifix} \Rightarrow \text{prefix/suffix}.$$

A code (resp., prefix code) $C \subset X^+$ is *maximal* (resp., *maximal prefix*) in $X$ if $C$ is not included by any other code (resp., prefix code) over $X$.

**Remark** A maximal and prefix code is clearly a maximal prefix code because it is not included in any code by the maximality. But a maximal prefix code is a prefix code, but is not necessarily a maximal code[2].

## 2.4 Petri Net Languages

Generally an automaton $\mathcal{A}$ over an alphabet $X$ consists of the (not necessarily finite) set $Q$ of states and the set $\mathcal{T} \subset Q \times X \times Q$ of labeled edges. The automaton $\mathcal{A}$ accepts a word $w = a_1 a_2 \dots a_n \in X^*$ which $(q_{i-1}, a_i, q_i) \in \mathcal{T} (1 \leq i \leq n)$, $q_0$ is in the set $I \subset Q$ of initial states, and $q_n$ is in the set $T \subset Q$ of final states. The language, denoted by $\mathcal{L}(\mathcal{A})$, is defined by all the words which are accepted by $\mathcal{A}$.

This concept can be applied to Petri nets as language acceptors. Roughly speaking, the markings of a Petri net correspond to the states of an automaton and the transitions induce the edges of an automaton. It comes to define a set of initial states, an alphabet, a set of final states.

### 2.4.1 Initial/Final Marking and Labeling Function

Roughly speaking, an initial marking and a final marking of a Petri net correspond to an initial state and a set of final states of an automaton in the role of language acceptance. A labeling function can be regarded as a morphism function from a free monoid $X^*$ to a free monoid $Y^*$, where $X$ is the set of transitions and $Y$ is an alphabet. Here we consider the variety of initial/final markings and labeling functions and give the definition of four types of Petri net languages depending of final markings. Each type have three variants by changing labeling functions.

*Initial Marking*

The initial state of a Petri net can be defined in several ways. The most common definition is to allow an arbitrary marking $\mu$ to be specified as an initial state. However, this definition can be modified in several ways. One convenient limitation is to restrict the initial state to a marking with one token in a start place and zero tokens elsewhere [22]. Another more general definition allows a set of initial markings rather than simply one marking. These three definitions are essentially the same. Certainly the start place definition is a special case of the initial marking definition, which is a special case of the set of initial markings definition. However, if a set of initial markings $M = \{\mu_1, \mu_2, \dots, \mu_k\}$ is needed with a start place definition, we can simply augment the Petri net with an extra place $p_0$ and a set of transitions $M = \{t'_1, t'_2, \dots, t'_k\}$. Transition $t_j$ would input a token from $p_0$ and would output marking $\mu_j$. Thus the behavior of the augmented net would be identical to a Petri net with a set of initial markings, except that each transition sequence would be preceded with $t_j$ if marking $t_j$ were used to start the execution.

We see then that these three definitions of start states for a Petri net are essentially equivalent. Out of a sense of tradition, we define a Petri net language as starting from a single arbitrary marking $\mu$.

*Labeling Function*

Labeling Petri net is another situation where several definition used. We must define both what the alphabet $X$ of a Petri net should be and how it is to be associated with the Petri net. We have indicated that the symbols are associated with the transitions, so that a sequence of transition firings generates a string of symbols of the language. The association of symbols to transitions is made by a *labeling function*, $\sigma : T \rightarrow X$. Variations in language definition may result from various restrictions placed on the labeling function.

A *free-labeled* Petri net is a labeled Petri net where all transitions are labeled distinctly, i.e., $\sigma(t) = \sigma(t')$ implies $t = t'$. The class of free Petri net languages is a subset of the class of free Petri net languages with a more general labeling function which does not require distinct labels. An even more general labeling function has been considered which allows null labeled transitions, $\sigma(t) = \lambda$. These $\lambda$-labeled transitions do not appear in a sentence of a Petri net language, and their occurrence in an execution of a Petri net thus goes unrecorded. These three classes of labeling functions(free, $\lambda$-free, and with $\lambda$-transitions) define three variations of Petri net languages.

*Final Marking*

The definition of final states for a Petri net has a major effect upon the language of a Petri net. Four major different definitions of the final state set of a Petri net have been suggested. Each of these may produce different Petri net Languages.

**DEFINITION 2.4.1** A language $L$ is an *L-type Petri net language* if there exist a Petri net $PN = (P, X, W, \mu_0)$, a labeling of the transitions $\sigma$, an initial marking $\mu$, and a finite set $F$ of final markings such that

$$L = \{\sigma(\beta) \,|\, \beta \in X^* \text{ and } \delta(\mu_0, \beta) \in F\}.$$

□

**DEFINITION 2.4.2** A language $L$ is an *G-type Petri net language* if there exist a Petri net $PN = (P, X, W, \mu_0)$, a labeling of the transitions $\sigma$, and a finite set $F$ of final markings such that

$$L = \{\sigma(\beta) \,|\, \beta \in X^* \text{ and there exists } \mu_f \in F \text{ such that } \delta(\mu_0, \beta) \geq \mu_f\}.$$

□

**DEFINITION 2.4.3** A language $L$ is an *T-type Petri net language* if there exist a Petri net $PN = (P, X, W, \mu_0)$, and a labeling of the transitions $\sigma$ such that

$$L = \{\sigma(\beta) \,|\, \beta \in X^* \text{ and } \delta(\mu_0, \beta) \neq \perp \text{ but for all } t_j \in X, \delta(\delta(\mu, \beta), t_j) = \perp\},$$

where $\perp$ denotes "undefined". □

**DEFINITION 2.4.4** A language $L$ is an *P-type Petri net language* if there exist a Petri net $PN = (P, X, W, \mu_0)$ and a labeling of the transitions $\sigma$ such that

$$L = \{\sigma(\beta) \mid \beta \in X^* \text{ and } \delta(\mu_0, \beta) \neq \perp\},$$

where $\perp$ denotes "undefined". □

### 2.4.2 Classes of Petri Net Languages

In addition to the four classes of languages based on different specifications of the final set, we have the following mentioned variations due to the labeling function. Table 2.1 lists twelve classes of languages which result from the cross product of the four types of final state specification and the three types of labeling functions. Each cell of Table 2.1 lists the notation which is used to denote each class of Petri net language.

Table 2.1: The classification of Petri net languages

| | Free | Non-$\lambda$ | $\lambda$-Transitions |
|---|---|---|---|
| $L$-type | $L^f$ | $L$ | $L^\lambda$ |
| $G$-type | $G^f$ | $G$ | $G^\lambda$ |
| $T$-type | $T^f$ | $T$ | $T^\lambda$ |
| $P$-type | $P^f$ | $P$ | $P^\lambda$ |

Figure 2.19 shows the inclusion relationships among the 12 classes of Petri net languages. An arc $A \longrightarrow B$ means that class $A$ include class $B$. A class with a free labeling is contained in a class with a $\lambda$-free labeling, and A class with a $\lambda$-free labeling is contained in a class with a $\lambda$-labeling. It is known that $G \subseteq L$, $G^\lambda \subseteq L^\lambda$, $T \subseteq L$, $T^\lambda = L^\lambda$[23].

### 2.4.3 Properties of Petri Net Languages

In Section 2.4.3 we limit ourselves here to considering only one class of Petri net languages, the $L$-type languages becasue this language has been investigated in the literature[40, 16, 22]. Some results have also been obtained by [16] for the prefix (P-type) languages.

Table 2.2 summarizes the closure properties of Petri net languages.

### 2.4.4 Additional Results

Most of the results presented here have been developed in both [40] and [16]. In addition, Hack has investigated a number of decidability problems for Petri net languages. The membership problem, whether a given string $\alpha$ is an element of a language $L(\gamma)$, is decidable, while the emptiness problem, whether the language $L(\gamma)$ is empty, can be easily seen to be equivalent to the reachability problem. The equivalence problem, whether two Petri net languages

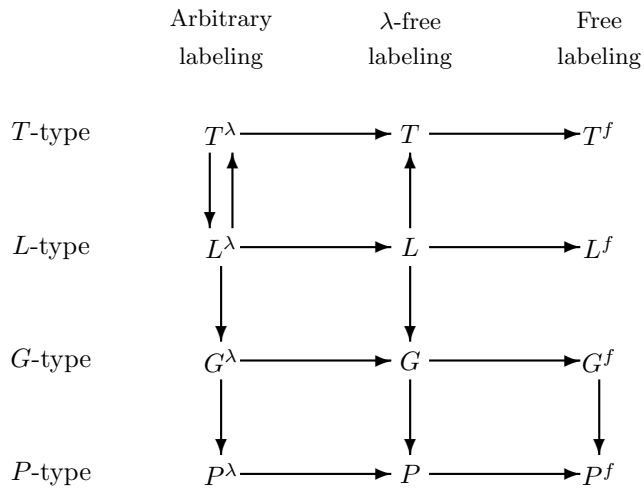



Figure 2.19: Inclusion relationship among the classes of Petri net languages

are equal, is undecidable. The inclusion problem, whether for given two Petri net languages one is contained within the other. Table 2.3 summarizes these results.

## 2.5 Petri Net Codes

G.Tanaka defined four types of prefix codes, called S-type, D-type, C-type and B-type Petri net codes, respectively, based on Petri nets[47]. Note that these codes is a Petri net language whose element is a firing sequence itself and labelling function is the identity mapping. Otherwise a obtained language

Table 2.2: Summary of closure properties of Petri net languages.[23]

| | $L^f$ | $L$ | $L^\lambda$ | $P^f$ | $P$ | $P^\lambda$ |
|---|---|---|---|---|---|---|
| Union | No | Yes | Yes | ? | Yes | Yes |
| Intersection | Yes | Yes | Yes | Yes | Yes | Yes |
| Concatenation | ? | Yes | Yes | ? | Yes | Yes |
| Concurrency | ? | Yes | Yes | ? | Yes | Yes |
| Regular substitution | No | λ-free | Yes | ? | Prefix | Prefix |
| Inverse homomorphism | Yes | Yes | Yes | Yes | Yes | Yes |
| Kleene star | ? | ? | No | ? | ? | ? |
| Complement | No | ? | No | No | ? | No |

Table 2.3: Summary of the decision properties of the L-type and P-type Petri net languages (D is decidable, U is undecidable).[23]

| | $L^f$ | $L$ | $L^\lambda$ | $P^f$ | $P$ | $P^\lambda$ |
|---|---|---|---|---|---|---|
| Membership | D | D | ? | D | D | D |
| Emptiness | ? | ? | ? | D | D | D |
| Finiteness | ? | ? | ? | D | D | D |
| Equvalence | ? | U | U | ? | U | U |
| Inclusion | ? | U | U | ? | U | U |

cannot form a code. In this section we explain their definitions and summarize fundamental properties of these codes.

**DEFINITION 2.5.1** Let $PN = (N, \mu_0) = (P, X, W, \mu_0)$ be a Petri net. The set

$$\mathrm{Stab}(\mu_0) = \{w \mid w \in L(\mu_0) \text{ and } \delta(\mu_0, w) = \mu_0\}$$

forms a free submonoid of $X^*$. The base of $\mathrm{Stab}(\mu_0)$, that is

$$(\mathrm{Stab}(\mu_0) \setminus \{1\}) \setminus (\mathrm{Stab}(\mu_0) \setminus \{1\})^2,$$

is said to be an *S-type Petri net code* (SPN code or SPNC, for short) if it is not empty, and denoted by $\mathbb{S}(PN)$ or $\mathbb{S}(N, \mu_0)$. □

Since $\mathbb{S}(PN, \mu_0)X^+ \cap \mathbb{S}(PN, \mu_0) = \emptyset$, $\mathbb{S}(PN, \mu_0)$ is a prefix code over $X$. The following set $\mathbb{D}(PN, \mu_0)$ is a subset of $\mathbb{S}(PN, \mu_0)$, so it is also a prefix codes.

**DEFINITION 2.5.2** Let $PN = (N, \mu_0) = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$. The set of all positive firing sequences of $\mathbb{S}(PN, \mu_0)$ is said to be a *D-type Petri net code* (DPN code or DPNC, for short), and denoted by $\mathbb{D}(PN)$ or $\mathbb{D}(N, \mu_0)$. □

.

**EXAMPLE 2.5.1** Let $PN = (P, T, W, \mu_0)$ be a Petri net where $\mu_0 = (2, 4)$ shown in Figure 2.20. Observing the reachability graph of $PN$, we obviously have $\mathrm{Stab}(\mu_0) = \{a^3b, a^2ba\}^*$ and $\mathbb{S}(PN) = \{a^3b, a^2ba\}$. An element in $\mathbb{S}(PN)$ is a nonempty minimal word in $\mathrm{Stab}(\mu_0)$ with respect to the prefix order $\leq_p$. Since an element in $\mathbb{D}(PN)$ is a positive firing sequence $\sigma$ in $\mathbb{S}(PN)$ which for any prefix $u$ of $\sigma$ $\delta(\mu_0, u)$ must be positive, $a^3b$ is in $\mathbb{D}(PN)$ but $a^2ba$ is not. Hence, $\mathbb{D}(PN) = \{a^3b\}$. □

**DEFINITION 2.5.3** Let $PN = (N, \mu_0) = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$. By $\mathbb{C}(PN)$ or $\mathbb{C}(N, \mu_0)$ denoted the set of all sequences $w \in L(\mu_0)$ satisfying the following conditions:

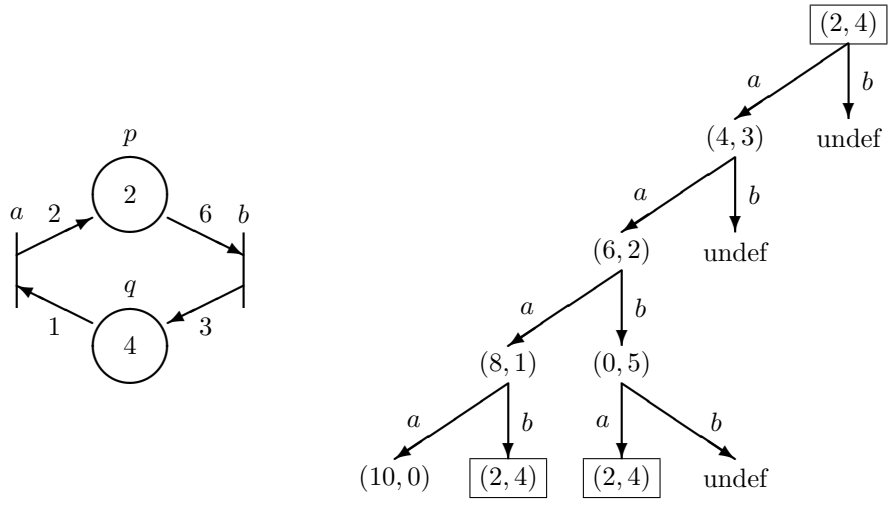



Figure 2.20: Petri net and its related codes

(1) $\delta(\mu_0, w)$ is not a positive marking, i.e., $w \in L(\mu_0) \setminus L_+(\mu_0)$.
(2) $\delta(\mu_0, v)$ is a positive marking for any proper prefix $v$ of $w$ i.e., $v \in L_+(\mu_0)$.
□

**DEFINITION 2.5.4** Let $PN = (N, \mu_0) = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$. By $\mathbb{B}(PN)$ or $\mathbb{B}(N, \mu_0)$ denoted the set of all sequences $w \in \mathbb{C}(PN, \mu_0)$ satisfying $\delta(\mu_0, v) \neq \mu_0$ for any prefix $v(\neq 1)$ of $w$ . □

.

By the definition 2.5.3, $\mathbb{C}(PN)$ is obviously a prefix code over $X$ if it is not empty. Since the set $\mathbb{B}(PN)$ is a subset of the prefix code $\mathbb{C}(PN)$, so that $\mathbb{B}(PN)$ is also a prefix code over $X$. Then $\mathbb{C}(PN)$ and $\mathbb{B}(PN)$ are said to be a *C-type Petri net code* (CPN code or CPNC, for short) and *B-type Petri net code* (BPN code or BPNC, for short), respectively.

The following proposition shows the fundamental relationship among a BPN code, a CPN code and a DPN code.

**PROPOSITION 2.5.1** [47]Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$. Then

$$\mathbb{C}(PN) = \mathbb{D}(PN)^*\mathbb{B}(PN).$$

□

**EXAMPLE 2.5.2** Again let $PN = (P, T, W, \mu_0)$ be a Petri net where $\mu_0 = (2, 4)$ shown in Figure 2.20. $\mathbb{C}(PN, \mu_0) = \{a^3b, a^2ba\}^*\{a^4, a^2b\}$. Noting that $\delta(\mu_0, a^3b) = \delta(\mu_0, a^2ba) = \mu_0$, Only elements $a^4$ and $a^2b$ in $\mathbb{C}(PN)$ are in $\mathbb{B}(PN)$, the others are not, where $\delta$ is the next-state function of the Petri net. □

The family of SPN codes (resp., DPN codes, CPN codes, BPN codes) is denoted by **SPNC** (resp., **DPNC**, **CPNC**, **BPNC**). The following inclusions are obvious.

$$\mathbf{DPNC} \subseteq \mathbf{SPNC} \quad \text{and} \quad \mathbf{BPNC} \subseteq \mathbf{CPNC}.$$

We will discuss the maximality of CPN codes in the following two chapters. Here we prepare the related terminologies and notations. A CPN code is said to be a *maximal C-type Petri net code* (maximal CPN code or mCPNC, for short) if it is a maximal prefix code. The family of all maximal CPN codes is denoted by **mCPNC**.

A CPN code $C$ is said to be a *input-ordinary C-type Petri net code* (input-ordinary CPN code or iCPNC, for short) if $C = \mathbb{C}(PN)$ for some input-ordinary Petri net $PN$. The family of all input-ordinary CPN codes is denoted by **iCPNC**.

Since an input-ordinary CPN code is clearly an maximal CPN code, we have the inclusion relation **iCPNC** $\subseteq$ **mCPNC**. The following problem remains open.

【Problem】 **mCPNC** $\subseteq$ **iCPNC** ?

Since it is too difficult to solve this problem in general Petri nets, in Chapter 5 we prove that the problem is solved affirmatively in some restricted Petri nets.

# Chapter 3

# Automorphism Groups of Nets

In this chapter we discuss the problem of automorphism groups of nets [52]. We construct a net called a transformation net from a transformation semigroup in a similar way to how we construct an automaton without outputs from a transformation semigroup[11]. It is well known that for a given group $G$ there exists an automaton such that its automorphism group is isomorphic to $G$. This fact is proved by using the property that the right regular representation of a group $G$ commutes with the left regular representation of $G$ as a permutation group on $G$. An analogous method is applied to prove our main result which states that We slightly touch on the generalization of the right regular representation of a group. That is, we construct a transformation net which corresponds to the right regular representation of a given group $G$ and we show that $\mathbf{Aut}(N)$ is isomorphic to $G$ by making some arc-weight of the net in certain conditions.

## 3.1 Transformation Nets

In this section, we define an automorphism of a net and a net called a transformation net. We represent a finite group by using some transformation net.

**DEFINITION 3.1.1** A net is a triple $(P, T, W)$ satisfying the following conditions (i) and (ii).

(i) $P$ and $T$ are finite nonempty sets with $P \cap T \neq \emptyset$. An element of $P$ (resp., $T$) is called a *place* (resp. a *transition*).

(ii) $W$: $(P \times T) \cup (T \times P) \to \{0, 1, 2, \cdots\}$ is called a *weight function*. Moreover, $a \in F$ iff $W(a) > 0$. □

The subset $F$ of $(P \times T) \cup (T \times P)$ is called the flow relation of a net $(P, T, W)$ if $F = \{(p, t) \in P \times T \mid W(p, t) > 0\} \cup \{(t, p) \in T \times P \mid W(t, p) > 0\}$. An element of $F$ is called an *arc*.

**DEFINITION 3.1.2** Let $N_1 = (P_1, T_1, W_1)$ and $N_2 = (P_2, T_2, W_2)$ be nets, and let $\alpha : P_1 \to P_2$ and $\beta : T_1 \to T_2$ be bijections. We define the mapping $(\alpha, \beta) : (P_1 \times T_1) \cup (T_1 \times P_1) \to (P_2 \times T_2) \cup (T_2 \times P_2)$ by

$$(\alpha, \beta)(a) = \begin{cases} (\alpha(p), \beta(t)) & \text{if} \quad a = (p, t) \in P_1 \times T_1, \\ (\beta(t), \alpha(p)) & \text{if} \quad a = (t, p) \in T_1 \times P_1. \end{cases}$$

If $W_2((\alpha, \beta)(a)) = W_1(a)$ for all $a \in (P_1 \times T_1) \cup (T_1 \times P_1)$, then $(\alpha, \beta)$ is called an *isomorphism* of $N_1$ onto $N_2$. An isomorhism is said to be an *automorphism* of $N_1$ if $N_1 = N_2$. □

The set $\mathbf{Aut}(N)$ of all automorphisms of a net $N = (P, T, W)$ is closed under the multiplication defined by

$$(\alpha_1, \beta_1) \cdot (\alpha_2, \beta_2) = (\alpha_1 \cdot \alpha_2, \beta_1 \cdot \beta_2).$$

Thus $\mathbf{Aut}(N)$ forms a group with the identity $(\mathbf{1}_P, \mathbf{1}_T)$, where $\mathbf{1}_P$ and $\mathbf{1}_T$ are the identity mappings of $P$ and $T$, respectively.

**DEFINITION 3.1.3** A net $N = (P, T, W)$ is said to be *transformation type* if

(i) $P$ is the union of nonempty sets $Q$ and $S$ with $Q \cap S = \emptyset$. We call the element of $Q$ (resp., $S$) an *inner*(resp., *source*) place.

(ii) For each $t \in T$,

$$S \cap t\bullet = \emptyset \quad \text{and} \quad |\bullet\, t \cap Q| = |\bullet\, t \cap S| = |Q \cap t\, \bullet| = 1,$$

where $|X|$ is the cardinality of a set $X$.

(iii) For each $(q, s) \in Q \times S$, there exists a unique $t \in T$ such that $\bullet t = \{q, s\}$. □

After this, a transformation type net is called *transformation net* simply.

Let $S$ be a transformation semigroup on a set $Q$, then we can define a transformation net $N = (Q \cup S, Q \times S, W)$, where $F$ is the flow relation of it and

$$\begin{aligned} F = \quad & \{(p, (p, s)) | p \in Q, s \in S\} \cup \{(s, (p, s)) | p \in Q, s \in S\} \cup \\ & \{((p, s), s(p)) | p \in Q, s \in S\}. \end{aligned}$$

Conversely, let $N = (Q \cup S, T, W)$ be a transformation net, and let $s \in S$ be a source place. For each $q \in Q$ there exists a unique $t \in T$ such that $(q, t), (s, t) \in F$. Therefore we can define a transformation $s'$ on $Q$ by $s' : q \mapsto p$, where $t\bullet = \{p\}$.

Let $G$ be a group. For any $y \in G$, $y'$ denotes the transformation defined by $y' : G \to G : x \mapsto xy$, and by $G'$ we denote the group $\{y' | y \in G\}$ with the multiplication of $y'$, $z' \in G'$ defined by $y' \cdot z'(x) = z'(y'(x))$. It is obvious that $G'$ is isomorphic to $G$.

**DEFINITION 3.1.4** Let $G$ be a finite group. A net $N = (G \cup G', G \times G', W)$ is called a *transformation net of* $G$ if It flow relation $F$ satisfies

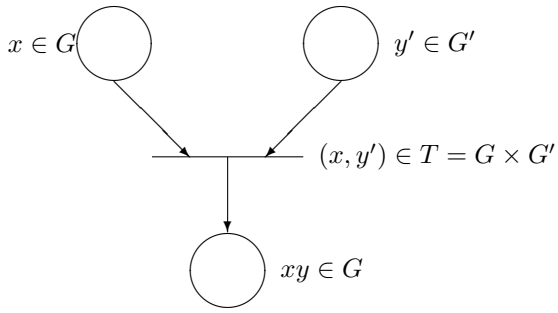



Figure 3.1: Structure of the transformation net of $G$

$$F = \{(x, (x,\, y'))|x \in G,\, y' \in G'\} \cup \{(y', (x,\, y'))|x \in G,\, y' \in G'\} \cup \{((x,\, y'), xy)|x, y \in G,\, y' \in G'\}.$$

□

Besides, no restriction placed on a weight function $W$. In Figure 3.1, we show the structure of the net representing $G$, where circles, a bar, and arrows denotes places, a transition, and arcs respectively.

## 3.2 Automorphism Groups of Nets

In this section we construct a transformation net which corresponds to the right regular representation of a given group $G$ and we show that for an arbitrarily finite group $G$ there exists a net $N$ such that $\mathbf{Aut}(N)$ is isomorphic to $G$ by introducing appropriate weights of arcs in the net.

First of all, it is easily verified that a set of all the automorphisms of a net $N = (P, T, W)$ is a group with the identity $(\mathbf{1}_P, \mathbf{1}_T)$, where $\mathbf{1}_P$ and $\mathbf{1}_T$ are identity mappings of $P$ and $T$, respectively.

**THEOREM 3.2.1** Let $G$ be a finite group and let $N_1 = (G \cup G', G \times G', W_1)$ be a transformation net of $G$, where the weight function $W_1$ is defined as follows:

$W_1$: For each $x \in G$ and each $y' \in G'$, $W_1(x, (x, y')) = W_1((x, y'), xy) = 1$ and $W_1(y', (x, y')) = \rho(y')$, where $\rho : G' \to \{2, 3, \cdots\}$ is injective.

Then the automorphism group $\mathbf{Aut}(N_1)$ of $N_1$ is isomorphic to $G$. □

(Proof) (i) First we show that for any $\varphi = (\alpha, \beta) \in \mathbf{Aut}(N_1)$ the restriction of $\alpha$ to $G'$ is the identity permutation of $G'$, i.e. $\alpha|G' = \mathbf{1}_{G'}$.

Let $\varphi = (\alpha,\, \beta) \in \mathbf{Aut}(N_1)$, then $\varphi(y', (x, y')) = (\alpha(y'), \beta(x, y'))$ and $W_1(\alpha(y'), \beta(x, y'))$ $= W_1(y', (x, y')) = \rho(y')$. Since $\rho$ is injective, the arc which has the weight $\rho(y')$ is the arc from $y'$. Therefore, we have $\alpha(y') = y'$.

Next we show that if $\alpha(e) = e$ for the identity $e \in G$, then $(\alpha,\, \beta) = (\mathbf{1}_P,\, \mathbf{1}_T)$, where $\mathbf{1}_P$ and $\mathbf{1}_T$ are the identity mappings of $P = G \cup G'$ and $T = G \times G'$, respectively.

Let $y' \in G'$ be an arbitrary source place and assume $\alpha(e) = e$, then

$$(\alpha, \beta)(e, (e, y')) = (\alpha(e), \beta(e, y')) = (e, \beta(e, y'))$$

and

$$(\alpha, \beta)(y', (e, y')) = (\alpha(y'), \beta(e, y')) = (y', \beta(e, y')).$$

Since $\beta(e, y')$ is the unique transition associated with $e$ and $y'$, we obtain $\beta(e, y') = (e, y')$.

Furthermore, since $W_1(\beta(e, y'), \alpha(y)) = W_1((e, y'), \alpha(y)) = 1$ and $(e, y')\bullet = \{y\}$, we have $\alpha(y) = ey = y$. Thus $\alpha = \mathbf{1}_P$.

Let $(x,\, y') \in G \times G'$, then $W_1(\alpha(x), \beta(x, y')) = W_1(x, \beta(x, y')) = 1$ and $W_1(\alpha(y'), \beta(x, y')) = W_1(y', \beta(x, y')) = 1$ Since $\beta(x, y')$ is the unique transition associated with $x$ and $y'$, we have $\beta(x, y') = (x, y')$, i.e. $\beta = \mathbf{1}_T$.

Finally, we show that there exists an isomorphism $\phi$ from $G$ onto $\mathbf{Aut}(N_1)$.

For each $g \in G$ we define the permutations $\alpha_g$ on $P = G \cup G'$ and $\beta_g$ on $G \times G'$ as follows:

$$\alpha_g(z) = \begin{cases} gz & \text{if } z \in G, \\ z & \text{if } z \in G', \end{cases} \tag{3.1}$$

$$\beta_g(x, y') = (gx, y'). \tag{3.2}$$

The permutation $\alpha_g$ applies each element $z$ of $G$ to multiplication of $g$ from the left, but never moves each element of $G'$.

Now, the mapping $(\alpha_g,\, \beta_g)$ maps

$$\begin{array}{ll} (x,\, (x,\, y')) \mapsto & (\alpha_g(x),\, \beta_g(x,\, y')) = (gx,\, (gx,\, y')), \\ ((x,\, y'),\, xy) \mapsto & (\beta_g(x,\, y'), \alpha_g(xy)) = ((gx,\, y'), gxy), \\ (y',\, (x,\, y')) \mapsto & (\alpha_g(y'),\, \beta_g(x,\, y')) = (y'\, (gx,\, y')), \end{array} \tag{3.3}$$

for arbitrary $x \in G$ and $y' \in G'$. Each arc corresponding to the first and the second rows in (3.3) has the weight of 1, Each arc corresponding to the third row has the weight of $\rho(y')$. The number of the arcs with positive weights in the net $N_1$ is of course finite $(3 \times |G|^2)$. So $(\alpha_g,\, \beta_g)$ preserves the weights of arcs. Therefore, $(\alpha_g,\, \beta_g) \in \mathbf{Aut}(N_1)$ holds. $\phi : g \longmapsto (\alpha_g,\, \beta_g)$ is a morphism from $G$ into $\mathbf{Aut}(N_1)$. Moreover, $\phi$ is injective since $\alpha_g(e) = \alpha_h(e)$ implies $g = h$.

We show that $\phi$ is surjective. Suppose that $\alpha(e) = g$. Since

$$\alpha_{g^{-1}}\alpha(e) = \alpha_{g^{-1}}(g) = g^{-1}g = e, \tag{3.4}$$

we have

$$(\alpha_{g^{-1}}\alpha,\, \beta_{g^{-1}}\beta) = \mathbf{1} \in \mathbf{Aut}(N_1). \tag{3.5}$$

The composition of $(\alpha,\, \beta)$ and $\phi(g^{-1}) = (\alpha_{g^{-1}},\, \beta_{g^{-1}}) \in \mathbf{Aut}(N_1)$ is the identity of $\mathbf{Aut}(N_1)$.

While, the definitions of $\alpha_g$ and $\beta_g$ yields

$$\begin{array}{rl} x \in G \Longrightarrow & \alpha_{g^{-1}}\alpha_g(x) = g^{-1}gx = x \\ y' \in G' \Longrightarrow & \alpha_{g^{-1}}\alpha_g(y') = \alpha_{g^{-1}}(y') = y' \end{array} \tag{3.6}$$

and

$$(x, y') \in G \times G' \Longrightarrow \beta_{g^{-1}}\beta_g(x, y') = (g^{-1}gx, y') = (x, y'). \tag{3.7}$$

Thus $(\alpha_{g^{-1}})^{-1} = \alpha_g$ and $(\beta_{g^{-1}})^{-1} = \beta_g$ hold. Therefore

$$\alpha = (\alpha_{g^{-1}})^{-1} = \alpha_g \text{ and } \beta = (\beta_{g^{-1}})^{-1} = \beta_g. \tag{3.8}$$

Hence, $\mathbf{Aut}(N_1) = \{(\alpha_g,\, \beta_g) |\, g \in G\}$. Since $\phi(g \cdot h) = (\alpha_{g\cdot h},\, \beta_{g\cdot h}) = (\alpha_g,\, \beta_h) \cdot (\alpha_g,\, \beta_h) = \phi(g) \cdot \phi(h)$, $\phi : G \to \mathbf{Aut}(N_1) : g \longmapsto (\alpha_g,\, \beta_g)$ is an isomorphism. It follows that $\mathbf{Aut}(N_1) \cong G$. □

The following THEOREM 3.2.2 is another expression of THEOREM 3.2.1.

**THEOREM 3.2.2** *For a given finite group $G$, there exists a net $N$ such that its automorphism group $\mathbf{Aut}(N)$ is isomorphic to $G$.* □

Since an automorphism group $\mathbf{Aut}(G)$ of a finite group $G$ is also a finite group, we have the following corollary by THEOREM 3.2.2. However, in the proof of the following corollary we construct another net in a different way shown in the proof of THEOREM 3.2.1.

The construction method of the net in the proof of COROLLARY 3.2.1 is effective for the case that $G$ is an automorphism group of some group $H$ and the order of $H$ is smaller than that of $G$, because we can construct the transformation net $N$ of $H$ which has fewer number of places and transitions than that of the transformation net of $G$.

**COROLLARY 3.2.1** For a finite group $G$ there exists a net $N$ such that its automorphism group $\mathbf{Aut}(N)$ is isomorphic to $\mathbf{Aut}(G)$. □

(Proof) Let $G$ be a finite group and let the net $N_2 = (G \cup G',\, G \times G',\, F,\, W_2)$ be a transformation net of $G$ defined by

$$W_2(a) = \begin{cases} 2 & \text{if } a = (e, (e, e')) \\ 1 & \text{if } a \neq (e, (e, e')). \end{cases} \tag{3.9}$$

where $e$ is an identity of $G$.

Define the mapping $\phi : \mathbf{Aut}(G) \to \mathbf{Aut}(N_2) : \tau \longmapsto (\alpha_\tau,\, \beta_\tau)$ by

$$\alpha_\tau(z) = \begin{cases} \tau(x) & \text{if } z = x \in G \\ \tau(y)' & \text{if } z = y' \in G', \end{cases} \tag{3.10}$$

$$\beta_\tau(x,\, y') = (\tau(x),\, \tau(y)'). \tag{3.11}$$

To prove that $\phi$ is an isomorphism, we show that (i) $(\alpha_\tau, \beta_\tau)$ is a element of $\mathbf{Aut}(N_2)$ , (ii) $\phi$ is surjective , and (iii) $\phi$ preserves the multiplication, respectively.

(i) We show that $\phi(\tau)$ is an automorphism of $N_2$.

It is obvious that $\alpha_\tau$ and $\beta_\tau$ are bijections on $P = G \cup G'$ and $T = G \times G'$, respectively.

For any $x \in G$ and $y' \in G'$, $(\alpha_\tau, \beta_\tau)$ moves $(x, (x, y'))$ to $(\tau(x), (\tau(x), \tau(y)'))$, $(y', (x, y'))$ to $(\tau(y)', (\tau(x), \tau(y)'))$, and $((x, y'), xy)$ to $((\tau(x), \tau(y)'), \tau(xy))$ respectively. Note that $\tau(xy) = \tau(x)\tau(y) = \tau(y)'(\tau(x))$, we find that $(\alpha_\tau, \beta_\tau) : F \to F$ is surjective. Moreover for any $a \in F$, if $a = (e, (e, e'))$, then $W_2((\alpha_\tau, \beta_\tau)(a)) = W_2(\tau(e), (\tau(e), \tau(e)')) = W_2(e, (e, e')) = W_2(a) = 2$ and if $a \neq (e, (e, e'))$, then $W_2((\alpha_\tau, \beta_\tau)(a)) = 1 = W_2(a)$. Hence, $\phi(\tau) = (\alpha_\tau, \beta_\tau)$ preserves the weight of each arc $a \in F$. This means that $\phi(\tau)$ is an automorphism of $N_2$. Thus, $\phi(\mathbf{Aut}(G)) \subset \mathbf{Aut}(N_2)$.

(ii) On the contrary, we show that for any $\varphi = (\alpha, \beta) \in \mathbf{Aut}(N_2)$ there exists some $\tau \in \mathbf{Aut}(G)$ such that $\varphi = \phi(\tau)$.

(ii)-1 Since only the arc $(e, (e, e'))$ has a weight of 2, we have $(\alpha(e), \beta(e, e')) = (e, (e, e'))$ must hold. Therefore,

$$\alpha(e) = e \quad \text{and} \quad \beta(e, e') = (e, e'). \tag{3.12}$$

Since $(\alpha(e'), \beta(e, e')) = (\alpha(e'), (e, e')) \in F$, $\alpha(e')$ is either $e$ or $e'$. While $\alpha(e) = e$ and $\alpha$ is bijective, we have $\alpha(e') = e'$.

(ii)-2 Each inner place $x \in G$ has $|G|$ input arcs and each source place $y' \in G'$ has no input arcs. Hence, $\alpha$ maps an inner place to some inner place, a source place to some source place, respectively. Therefore, some bijective morphism $\gamma : G \to G$ exists such that

$$\forall y \in G,\ \alpha(y') = \gamma(y)'. \tag{3.13}$$

Strictly speaking, $\gamma = {}'^{-1} \cdot (\alpha|G') \cdot '$.

(ii)-3 $\varphi$ maps the arc $(x, (x, y'))$ to $(\alpha(x), \beta(x, y'))$ and the arc $(y', (x, y'))$ to $(\alpha(y'), \beta(x, y'))$, respectively. This means that $\beta(x, y')$ is the unique transition associated with $\alpha(x)$ and $\alpha(y')$. Therefore, we have

$$\beta(x, y') = (\alpha(x), \alpha(y')) = (\alpha(x), \gamma(y)'). \tag{3.14}$$

While the only one arc go out from the above transition $\beta(x, y')$ to the inner place $\alpha(xy)$, by the definition of a transformation net, we have

$$\alpha(xy) = \alpha(x)\gamma(y). \tag{3.15}$$

Take $x = e$, then we have $\gamma(y) = \alpha(y)$. By substituting $\alpha(y)$ for $\gamma(y)$ in (3.15),

$$\alpha(xy) = \alpha(x)\alpha(y). \tag{3.16}$$

Since $\alpha$ is injective (surjective) morphism, the restriction $\alpha|G$ of $\alpha$ on $G$ is an automorphism of $G$. We denote this automorphism by $\tau$.

(ii)-4 $\alpha(x') = \gamma(x)' = \alpha(x)' = \tau(x)'$ and $\beta(x, y) = (\alpha(x), \gamma(y)') = (\alpha(x), \alpha(y)') = (\tau(x), \tau(x)')$. So $\varphi = \phi(\tau) \in \phi(\mathbf{Aut}(G))$.

(iii) Finally we show that $\phi(\tau\cdot\sigma) = \phi(\tau)\cdot\phi(\sigma)$ holds for any $\tau$, $\sigma \in \mathbf{Aut}(G)$.

The permutation $\alpha_{\tau\cdot\sigma}$ on $G \cup G'$ identifies the composition $\alpha_\tau \cdot \alpha_\sigma$ of $\alpha_\tau$ and $\alpha_\sigma$, because $\alpha_{\tau\cdot\sigma}(x) = \tau \cdot \sigma(x) = \tau(\sigma(x)) = \alpha_\tau \cdot \alpha_\sigma(x)$ for any $x \in G$, and $\alpha_{\tau\cdot\sigma}(y') = (\tau\cdot\sigma(y))' = (\tau(\sigma(y)))' = \alpha_\tau(\sigma(y)') = \alpha_\tau(\alpha_\sigma(y')) = \alpha_\tau\cdot\alpha_\sigma(y')$ for any $y' \in G'$. The permutation $\beta_{\tau\cdot\sigma}$ on $G \times G'$ identifies the composition $\beta_\tau \cdot \beta_\sigma$ of $\beta_\tau$ and $\beta_\sigma$, because for any $(x, y') \in G \times G'$, $\beta_\tau \cdot \beta_\sigma(x, y') = \beta_\tau(\beta_\sigma(x, y'))$ $= \beta_\tau(\sigma(x), \sigma(y)') = (\tau(\sigma(x)), \tau(\sigma(y))') = (\tau \cdot \sigma(x)), \tau \cdot \sigma(y)') = \beta_{\tau\cdot\sigma}(x, y')$.

Hence (3.17) holds.

$$\begin{aligned} \phi(\tau \cdot \sigma) &= (\alpha_{\tau\cdot\sigma}, \beta_{\tau\cdot\sigma}) \\ &= (\alpha_\tau \cdot \alpha_\sigma, \beta_\tau \cdot \beta_\sigma) \\ &= (\alpha_\tau, \beta_\tau) \cdot (\alpha_\sigma, \beta_\sigma) \\ &= \phi(\tau) \cdot \phi(\sigma). \end{aligned} \tag{3.17}$$

By above (i), (ii) and (iii), we have shown that $\phi$ is an isomorphism form $\mathbf{Aut}(G)$ to $\mathbf{Aut}(N_2)$, so that $\mathbf{Aut}(N) \cong \mathbf{Aut}(G)$. □

## 3.3 Remarks and Further Works

The notion of the automorphism group $\mathbf{Aut}(N)$ of a Petri net structure $N$ is newly introduced . We show the main theorem that for a given finite group $G$ there exists a Petri net structure $N$, called a transformation net, such that $\mathbf{Aut}(N)$ is isomorphic to $G$. The structure $N$ corresponds to the right regular representation of $G$.

A transformation net is only a Petri net structure without marking. We can consider a behavior of a Petri net by adding a marking to it. I have had an equivalent condition to the reachability problem with respect to a transformation net $N = (G \cup G', G \times G', W)$, where $G$ is the residue group of order $n$ [28].

Let $\mu$ and $\lambda$ be markings with $\lambda(g') = 0$ for any $g' \in G'$ (we may assume this condition without loss of generality). Then if $\mu$ is reachable from $\lambda$, then

$$(1)\ \sum_{g\in G} \mu(g) = \sum_{g\in G} \lambda(g),$$

$$(2)\ \sum_{g\in G} g \cdot (\mu(g) + \mu(g')) \equiv \sum_{g\in G} g \cdot \lambda(g) \pmod{n}.$$

Though this is just a necessary condition for the reachability. we can check the conditions (1) and (2) in the order of $n$, where $n = |G|$.

The remaining problem is the reverse implication, that is, whether both (1) and (2) imply that $\mu$ is reachable from $\lambda$ or not.

# Chapter 4

# Properties of CPN Codes

In this chapter, we consider the language over an alphabet $X$ generated by a given Petri net with a positive initial marking, called a *CPN code*. This code becomes a prefix code over $X$. We are interested in CPN codes which are maximal prefix codes, called *maximal CPN codes* over $X$. We will investigate various properties of maximal CPN codes. Moreover, we will prove that a CPN code is a context-sensitive language in two different ways.

## 4.1 Maximal CPN Codes of the Form $C^n$

Let $D = (P, X, W, \mu_0)$ be a Petri net with an initial marking $\mu_0$ where $P$ is the set of places, $X$ is the set of transitions, $W$ is the weihgt function and $\mu_0 \in N^P$ is a positive marking, i.e. $\mu_0(p) > 0$ for any $p \in P$. Notice that $\mu_0(p)$ is meant the number of tokens at $p$ of the marking $\mu_0$.

Let $\delta$ be the next-state function of $D$. A language $C$ is called a *CPN code* over $X$ generated by $D$ and denoted by $C = \mathbb{C}(D)$ if $C = \{u \in X^+ \mid \exists p \in P, \delta(\mu_0, u)(p) = 0, \forall q \in P, \delta(\mu_0, u)(q) \geq 0$, and $\forall q' \in P, \delta(\mu_0, u')(q') > 0$ for $u' \in P_r(u) \setminus \{u\}$ where $P_r(u)$ is the set of all prefixes of $u\}$. Then it is obvious that $C = \mathbb{C}(D)$ is a prefix code over $X$ if $C = \mathbb{C}(D) \neq \emptyset$. Notice that $CPN$ codes were introduced in [47]. If $C$ is a maximal prefix code, then $C$ is called an *maximal CPN codes* over $X$. Now let $u = a_1 a_2 \dots a_r \in X^*$ where $a_i \in X$. Then, for any $p \in P$, by $p(u)$ we denote $-\Delta(u)(p)$, that is, the negative value at the place $p$ of the displacement $\Delta(u)$ of $u$, which is the amount of consumed tokens at $p$ by the firing sequence $u$.

**LEMMA 4.1.1** Let $C = \mathbb{C}(D)$ be a finite maximal CPN code where $D = (P, X, W, \mu_0)$. By $t_p$ we denote $\mu_0(p)$ for any $p \in P$. For any $u, v \in C$, if there exists a $p \in P$ such that $t_p = p(u) = p(v)$, then $C$ is a full uniform code over $X$, i.e. $C = X^n$ for some $n, n \geq 1$.

(Proof) Let $a_1 a_2 \dots a_{n-1} a_n \in C$ be a word with the longest length in $C$. Since $C$ is a finite maximal prefix code over $X$, $a_1 a_2 \dots a_{n-1} X \subseteq C$. Assume $a_1 a_2 \dots a_{i-1} a_i \ X^{n-i} \subseteq C$. Suppose $a_1 a_2 \dots a_{i-1} b\alpha \in C$ where $a_i \neq$

$b \in X$ and $\alpha \in X^*$ with $|\alpha| < n - i$. Then there exists $\beta \in X^*$ such that $a_1a_2 \dots a_{i-1}a_i\alpha\beta a_i \in C$. Notice that $|\alpha\beta| = n-i-1$. Let $u = a_1a_2 \dots a_{i-1}a_i\alpha\beta a_i$ and $v = a_1a_2 \dots a_{i-1}b\alpha$. By assumption, there exists $p \in P$ such that $t_p = p(u) = p(v)$. Consider $w = a_1a_2 \dots a_{i-1}a_ib\alpha\beta \in C$. Since $p(a_1a_2 \dots a_{i-1}a_i\alpha\beta a_i) = t_p, p(a_1a_2 \dots a_{i-1}a_i) > p(a_1a_2 \dots a_{i-1})$. On the other hand, notice that $t_p = p(a_1a_2 \dots a_{i-1}b\alpha)$. This means that $\delta(\delta(\mu_0, a_1a_2 \dots a_{i-1}), b\alpha))$ can be computed only when at least $t_p - p(a_1a_2 \dots a_{i-1})$ tokens are placed at $p \in P$, where $\delta$ is the next-state function of $D$. Consequently, this contradicts the fact $p(w) = p(a_1a_2 \dots a_{i-1}a_ib\alpha\beta) \leq t_p$. Hence $a_1a_2 \dots a_{i-1}bX^{n-i} \subseteq C$. This yeilds $a_1a_2 \dots a_{i-1}X^{n-i+1} \subseteq C$. By induction, we have $X^n \subseteq C$ and $X^n = C$. □

**LEMMA 4.1.2** Let $A, B$ be finite maximal prefix codes over $X$. If $AB$ is a maximal CPN code over $X$, then $B$ is a maximal CPN code over $X$.

(Proof) Let $AB = \mathbb{C}(D)$ where $D = (P, X, W, \mu_0)$ with the next-state function $\delta$. Let $u \in A$ and let $D' = (P, X, W, \delta(\mu_0, u))$. Then obviously $\delta(\mu_0, u) \in N^P$ and $B \subseteq \mathbb{C}(D')$. Since $B$ is a maximal prefix code over $X$, $B = \mathbb{C}(D')$, i.e. $B$ is a maximal CPN code over $X$. □

**COROLLARY 4.1.1** Let $C^n$ be a finite maximal CPN code over $X$ for some $n, n \geq 2$. Then $C^k$ is a maximal CPN code over $X$ for any $k, 1 \leq k \leq n$. □

Now we provide a fundamental result.

**PROPOSITION 4.1.1** Let $C^n$ be a finite maximal CPN code over $X$ for some $n, n \geq 2$. Then $C$ is a full uniform code over $X$.

(Proof) By COROLLARY 4.1.1, we can assume that $n = 2$. Let $D = (P, X, W, \mu_0)$ be a Petri net such that $C^2 = \mathbb{C}(D)$. Let $t_p = \mu_0(p)$ for any $p \in P$. Since $u^2 \in C^2$, there exists $p \in P$ such that $t_p = p(u^2) = 2p(u)$. Let $s_p = t_p/2$ for such a $p \in P$. Then $s_p$ can be well-defined. For remaining $q \in P$, we define $s_q = t_q$. Let define $\mu_0'$ as $\mu_0'(p) = s_p$ for any $p \in P$. First we check whether $\delta(\mu_0', u)$ is computable if $p(u) = s_p = t_p/2$. The problem is whether the number of tokens at each place $p \in P$ is enough to make fire the sequence of transitions $u$. By the proof of LEMMA 4.1.2, we have at most $|C|$ Petri nets recognizing $C$. Let $C = \{u_i \mid i = 1, 2, \dots, r\}$ and let $D_i = (P, X, W, \mu_{0,i})$ where $\mu_{0,i} = \delta(\mu_0, u_i), i = 1, 2, \dots, r$, and $\delta$ is the next-state function of $D$. Notice that $\mathbb{C}(D_i) = C$ for any $i, i = 1, 2, \dots, r$. Since $u_i^2 \in C^2$ for any $i, i = 1, 2, \dots, r$, there exists $p \in P$ such that $p(u_i^2) = t_p$, i.e. $p(u_i) = t_p/2$. Moreover, notice that, for another $u_j \in C$, $p(u_j) \leq t_p/2$ because $p(u_j^2) \leq t_p$. Hence $min\{\mu_{0,i}(p)\} = s_p$ if $s_p = t_p/2$. Since $\mathbb{C}(D_i) = C$ for any $i, i = 1, 2, \dots, r$, it follows that the marking $\mu_0'$ with $\mu_0'(p) = s_p, p \in P$ has enough tokens at each place to make fire the sequence of transitions $u \in C$. Let $D' = (P, X, W, \mu_0')$. Then $C \subseteq \mathbb{C}(D')$. On the other hand, since $C$ is a maximal prefix code over $X$, $C = \mathbb{C}(D')$. Let $u, v \in C$. Since $uv \in C^2$, there exists $p \in P$ such that $t_p = p(uv)$. Notice that $p(uv) = p(u) + p(v)$. If $p(u) \neq p(v)$, then $p(uv) < max\{p(uu), p(vv)\}$. However, since $uu, vv \in C^2$, $max\{p(uu), p(vv)\} \leq t_p$, a contradiction. Hence $s_p = p(u) = p(v)$. By LEMMA 4.1.1, $C$ is a full uniform code over $X$. □

**THEOREM 4.1.1** The property being a maximal CPN code over $X$ is not preserved under concatenation.

(Proof) Let $C \subseteq X^+$ be a maximal CPN code over $X$ which is not a full uniform code over $X$. Suppose $C^2$ is a maximal CPN code over $X$. Then, by PROPOSITION 4.1.1, $C$ is a full uniform code over $X$, a contradiction. □

**REMARK 4.1.1** We can prove the above corollary in a different way. Let $X = \{a, b\}$, let $A = \{a, ba, bb\}$ and let $B = \{b, ab, aa\}$. Then $ab, aaa, bbb \in AB$ and $|aaa|, |bbb| > |ab|$. By the following lemma, $AB$ is not a maximal CPN code over $\{a, b\}$.

**LEMMA 4.1.3** Let $C \subseteq X^+$ be a maximal CPN code over $X$. Then there exists $a \in X$ such that $a^{min\{|u||u\in C\}} \in C$.

(Proof) Let $D = (P, X, W, \mu_0)$ be a Petri net with $C = \mathbb{C}(D)$. Moreover, let $v \in C$ such that $|v| = min\{|u| \,|\, u \in C\}$. Let $v = v'a^i$ where $i \geq 1, v' \in X^*, a \in X$ and $v' \notin X^*a$. If $v' = \lambda$, then we have done. Let $v' = v''b$, i.e. $v = v''ba^i$ where $b \in X$ and $b \neq a$. Since $v \in C$, there exists $p \in P$ such that $p(v) = t_p = \mu_0(p)$. If $p(a) \geq p(b)$, then we consider $p(v''aa^i)$. If $p(a) > p(b)$, then $p(v''aa^i) > t_p$ and some proper prefix of $v''aa^i$ must be an element of $C$. However, this contradicts the assumption that $v$ has the minimum length as a word in $C$. Therefore, $p(a) = p(b)$ and $p(v''aa^i) = t_p$. This implies that $v''aa^i \in C$. Now let $p(b) > p(a)$. In this case, we consider $v''bb^i \in X^+$. It is obvious that $p(v''bb^i) > p(v) = t_p$. This contradicts again that the assumption that $v$ has the minimum length as a words in $C$. Hence $v''a^{i+1} \in C$ and $|v''a^{i+1}| = |v|$. Contuining the same procedure, we have $a^{|v|} \in C$. This completes the proof of the lemma. □

**REMARK 4.1.2** If $C$ is an infinite maximal CPN code over $X$, then PROPOSITION 4.1.1 does not hold true. For instance, let $X = \{a, b\}$ and let $C = b^*a$. Then both $C$ and $C^2 = b^*ab^*a$ are maximal CPN codes over $X$. □

In the following section, we will generalize PROPOSITION 4.1.1. □

## 4.2 Maximal CPN Codes of the Form $AB$

In fact, we can generalize PROPOSITION 4.1.1 as follows:

**PROPOSITION 4.2.1** Let $A, B$ be finite maximal prefix codes over $X$. If $AB$ is a maximal CPN code over $X$, then $A$ and $B$ are full uniform codes over $X$.

(Proof) (i) Let $a_1a_2 \ldots a_{n-1}a_n$ be one of the longest words in $A$. Since $A$ is a finite maximal prefix code over $X$, $a_1a_2 \ldots a_{n-1}X \subseteq A$. Assume that $a_1a_2 \ldots a_{i-1}a_iX^{n-i} \subseteq A$ where $1 \leq i < n$ and $a_1a_2 \ldots a_{i-1}X^{n-i+1} \setminus A \neq \emptyset$. Then there exist $b \in X$ and $\alpha \in X^*$ such that $b \neq a_i$, $|\alpha| < n - i$ and $a_1a_2 \ldots a_{i-1}b\alpha \in A$. Consider the word $a_1a_2 \ldots a_{i-1}a_ib\alpha a_i^t \in A$ where

$t \geq 0$ and $|\alpha| + t = n - i - 1$. Since $B$ is a finite maximal prefix code over $X$, there exists $h, h \geq 1$ such that $a_i^h \in B$. Hence $a_1 a_2 \dots a_{i-1} b \alpha a_i^h \in AB$ and $a_1 a_2 \dots a_{i-1} a_i b \alpha a_i^t a_i^h \in AB$. Then there exists $p \in P$ such that $p(a_1 a_2 \dots a_{i-1} b \alpha a_i^h) = t_p$ and $p(a_i) > 0$. It is obvious that $p(a_1 a_2 \dots a_{i-1} a_i b \alpha a_i^t a_i^h) = (t+1)p(a_i) + t_p > t_p$, a contradiction. Therefore, $a_1 a_2 \dots a_{i-1} X^{n-i+1} \subseteq A$. Continuing this procedure, we have $X^n \subseteq A$ and hence $X^n = A$.
(ii) Let $b_1 b_2 \dots b_{m-1} b_m$ be one of the longest words in $B$. Assume that $b_1 b_2 \dots b_{j-1} b_j X^{m-j} \subseteq B$ where $1 \leq j < m$ and $b_1 b_2 \dots b_{j-1} X^{m-j+1} \setminus B \neq \emptyset$. Then there exist $c \in X$ and $\beta \in X^*$ such that $c \neq b_j$, $|\beta| < m - j$ and $b_1 b_2 \dots b_{j-1} c \beta \in B$. Let $c\beta \in X^* d$ where $d \in X$. By the above assumption, $b_1 b_2 \dots b_{j-1} b_j c \beta d^s \in B$ where $s \geq 0$. By (i), $A = X^n$ for some $n, n \geq 1$. Hence $(d^{n-1} b_j) b_1 b_2 \dots b_{j-1} c \beta \in AB$ and $d^n b_1 b_2 \dots b_{j-1} b_j c \beta d^s \in AB$. Notice that there exists $q \in P$ such that $t_q = q((d^{n-1} b_j) b_1 b_2 \dots b_{j-1} c \beta)$ and $q(d) > 0$. Then $q(d^n b_1 b_2 \dots b_j c \beta d^s) = t_q + (s+1) q(d) > t_q$, a contradiction. Hence $b_1 b_2 \dots b_{j-1} X^{m-j+1} \subseteq B$. Continuing this procedure, we have $X^m \subseteq B$ and hence $X^m = B$. This completes the proof of the proposition. □

## 4.3 Constructions of Maximal CPN Codes

In this section, we provide two construction methods of maximal CPN codes.

**DEFINITION 4.3.1** Let $A, B \subseteq X^+$. Then by $A \oplus B$ we denote the language $(\cup_{b \in X} \{(P_r(A) \setminus A) \diamond B b^{-1}\} b) \cup (\cup_{a \in X} \{(P_r(B) \setminus B) \diamond A a^{-1}\} a)$ where $\diamond$ is meant the shuffle operation and $Ca^{-1} = \{u \in X^+ | ua \in C\}$ for $C \subseteq X^+$ and $a \in X$. □

**PROPOSITION 4.3.1** Let $X = X_1 \cup X_2$ where $X_1, X_2 \neq \emptyset, X_1 \cap X_2 = \emptyset$. If $A \subseteq X_1^+$ is a maximal CPN code over $X_1$ and $B \subseteq X_2^+$ is a maximal CPN code over $X_2$, then $A \oplus B$ is a maximal CPN code over $X$. □

(Proof) Let $D_1 = (P_1, X_1, W_1, \mu_1)$ and $D_2 = (P_2, X_2, \delta_2, \mu_2)$ be Petri nets such that $\mathbb{C}(D_1) = A$ and $\mathbb{C}(D_2) = B$, respectively, and $P_1 \cap P_2 = \emptyset$. Now we define the following Petri net $D$: $D = (P_1 \cup P_2, X, \delta, \mu_0)$ where $\mu_0(p) = \mu_1(p)$ for $p \in P_1$ and $\mu_0(q) = \mu_2(q)$ for $q \in P_2$, and $W(p, a) = W_i(p, a)$ for any $(p, a) \in P_i \times X_i$ and $W(a, p) = W_i(a, p)$ for any $(a, p) \in X_i \times P_i$ $(i = 1, 2)$. Then $A \oplus B = \mathbb{C}(D)$. □

**EXAMPLE 4.3.1** Let $X = \{a, b\}$. Consider $A = \{a\}$ and $B = \{bb\}$. Then both $A$ and $B$ are maximal CPN codes over $\{a\}$ and $\{b\}$, respectively. Hence $A \oplus B = \{a, ba, bb\}$ is a maximal CPN codes over $X$. □

**PROPOSITION 4.3.2** Let $A, B \subseteq X^+$ be finite maximal CPN codes over $X$. Then $A \oplus B$ is a maximal CPN codes over $X$ if and only if $A = B = X$.

(Proof) ($\Leftarrow$) Since $A \oplus B = X \oplus X = X$, $A \oplus B$ is a maximal CPN codes over $X$.
($\Rightarrow$) Assume $A \neq X$. Then there exists $a \in X$ such that $a^i \in A, i \geq 2$. Moreover, since $B$ is a maximal CPN code over $X$, there exists some $j, j \geq 1$

such that $a^j \in B$. Notice that $\lambda, a \in P_r(a^i) \setminus \{a^i\}$. Hence $a^j, a^{j+1} \in A \oplus B$. This implies that $A \oplus B$ is not a prefix code. Consequently, $A \oplus B$ is not even a CPN code. The proof can be done in the same way for the case $B \neq X$. □

**REMARK 4.3.1** For the class of infinite maximal CPN codes over $X$, the situation is different. For instance, let $X = \{a, b\}$ and let $A = B = b^*a$. Then $A \oplus B = b^*a$ and $A, B$ and $A \oplus B$ are maximal CPN codes over $X$.

**PROPOSITION 4.3.3** Let $A, B \subseteq X^+$ be maximal CPN codes over $X$. Then there exist an alphabet $Y$, a maximal CPN code $D \subseteq Y^+$ over $Y$, a $\lambda$-free homomorphism $h$ of $Y^*$ onto $X^*$ such that $A \oplus B = h(D)$.

(Proof) Let $X' = \{a' \mid a \in X\}$. For $u = a_1 a_2 \dots a_n \in X^n$, we denote $u' = a'_1 a'_2 \dots a'_n \in X'^n$. Let $Y = X \cup X'$ and let $h$ be the homomorphism of $Y^*$ onto $X^*$ defined as: $h(a) = a, h(a') = a$ for any $a \in X$. Let $B' = \{u' \mid u \in B\} \subseteq X'^+$. Then $B'$ is a maxial CPN code over $X'$. By PROPOSITION 4.3.1, $D = A \oplus B'$ is a maximal CPN code over $Y$ and $A \oplus B = h(D)$. □

**THEOREM 4.3.1** The property being a maximal CPN code over $X$ is not preserved under $\lambda$-free homomorphism. □

**LEMMA 4.3.1** Let $C \subseteq X^+$ be a maximal CPN code over $X$ and let $a, b \in X$. If $bbaa \in C$, then $baba \in C$. □

(Proof) Let $D = (P, X, W, \mu_0)$ be a Petri net with $C = \mathbb{C}(D)$. Since $bbaa \in C$, there exists $p \in P$ such that $p(bbaa) = t_p = \mu_0(p)$. It follows from the assumption that $C$ is a maximal CPN code over $X$ that $p(bbaa) = p(baba) = t_p$. Since $p(b), p(bb), p(bba) \leq t_p$, we have $p(b), p(ba) \leq t_p$ and $p(bab) \leq t_p$. Moreover, since $q(b), q(bb), q(bba), q(bbaa) \leq t_q$ for any $q \in P$, we have $q(b), q(ba), q(bab) \leq t_q$ and $q(baba) \leq t_q$. Hence $baba \in C$. □

**REMARK 4.3.2** By the above lemma, a maximal prefix code over $X$ having the property in LEMMA 4.3.1 cannot be necessarily realized by a Petri net. For instance, let $X = \{a, b\}$ and let $C = \{a, ba, bbaa, bbab, bbb\}$. Then $C$ is a maximal prefix code over $X$. However, by LEMMA 4.3.1, it is not a maximal CPN code over $X$. □

Now we introduce another method to construct maximal CPN codes.

**DEFINITION 4.3.2** Let $A \subseteq X^+$. By $m(A)$, we denote the language $\{v \in A \mid \forall u, v \in A, \forall x \in X^*, v = ux \Rightarrow x = 1\}$. Obviously, $m(A)$ is a prefix code over $X$. Let $A, B \subseteq X^+$. By $A \otimes B$, we denote the language $m(A \cup B)$. □

**THEOREM 4.3.2** Let $A, B$ be maximal CPN codes over $X$. Then, $A \otimes B$ is a maximal CPN code over $X$.

(Proof) Let $D = (P, X, W_D, \mu_D)$ and $E = (Q, X, W_E, \mu_E)$ be Petri nets with a positive initial markings such that $A = \mathbb{C}(D)$, $B = \mathbb{C}(E)$ and $P \cap Q = \emptyset$.

Let $D \otimes E = (P \cup Q, X, W_D \times W_E, \mu_D \times \mu_E)$ where $W_D \times W_E(p) = W_D(p)$ and $\mu_D \times \mu_E(p) = \mu_D(p)$ for $p \in P$, and $W_D \times W_E(q) = W_E(p)$ and $\mu_D \times \mu_E(q) = \mu_D(q)$ for $q \in Q$. Let $\delta_D$, $\delta_E$ and $\delta$ be the next-state functions of $D$, $E$ and $D \otimes E$, respectively.

We will show that $\mathbb{C}(D \otimes E) = A \otimes B$. First, we show that $A \otimes B$ is a maximal prefix codes over $X$. Let $u \in X^*$. Since $A$ is a maximal prefix code over $X$, $u \in AX^*$ or $uX^* \cap A \neq \emptyset$. If $u \in AX^*$, then $m(A \cup B)X^* = (A \otimes B)X^*$. Assume $uX^* \cap A \neq \emptyset$. Since $B$ is a maximal prefix code over $X$, $uX^* \cap B \neq \emptyset$ or $u \in BX^*$. Then either $uX^* \cap (A \otimes B) \neq \emptyset$ or $u \in (A \otimes B)X^*$ hold. Hence $A \otimes B$ is a maximal prefix code over $X$.

Now we show that $\mathbb{C}(D \otimes E) = A \otimes B$. Let $u \in A \otimes B$. Then $u \in A \cup B$. Assume $u \in A$. Then there exists $p \in P$ such that $\delta_D(\mu_D, u)(p) = 0$. Moreover, $\delta_D(\mu_D, u)(p') \geq 0$ and $\delta_D(\mu_D, u')(p') > 0$ for any $p' \in P$ and $u' \in P_r(u) \setminus \{u\}$. On the other hand, since $u \in A \otimes B$ and $u \in A$, there exists $x \in X^*$ such that $ux \in B$. Hence, for any $q \in Q$, $\delta(\mu_E, u)(q) \geq 0$ and $\delta_E(\mu_E, u')) > 0$ for any $q \in Q$ and $u' \in P_r(u) \setminus \{u\}$. This means that $\delta(\mu_D \times \mu_E, u)(p) = 0$ for $p \in P \cup Q$ and $\delta(\mu_D \times \mu_E, u)(r) \geq 0$, $\delta(\mu_D \times \mu_E, u')(r) > 0$ for any $r \in P \cup Q$ and $u' \in P_r(u) \setminus \{u\}$. Hence $u \in \mathbb{C}(D \otimes E)$. By the same way, we can prove that $u \in \mathbb{C}(D \otimes E)$ for $u \in B$. Therefore, $A \otimes B \subseteq \mathbb{C}(D \otimes E)$. By the maximality of $A \otimes B$, we have $A \otimes B = \mathbb{C}(D \otimes E)$. □

**EXAMPLE 4.3.2** It is obvious that $a^*b$ and $(a \cup b)^3$ are maximal CPN codes over $\{a, b\}$. Hence $a^*b \otimes (a \cup b)^3 = \{b, ab, aaa, aab\}$ is a maximal CPN code over $\{a, b\}$.

**REMARK 4.3.3** THEOREM 4.3.2 does not hold for the class of CPN codes over $X$. The reason is the following: Suppose that $A \otimes B$ is a CPN code over $X$ for any two CPN codes $A$ and $B$ over $X$. Then we can show that, for a given finite CPN code $A$ over $X$, there exists a finite maximal CPN code $B$ over $X$ such that $A \subseteq B$ as follows. Let $A \subseteq X^+$ be a finite CPN code over $X$ which is not a maximal CPN code. Let $n = max\{|u| \,|\, u \in A\}$. Consider $X^n$ which is a maximal CPN code over $X$. By assumption, $A \otimes X^n$ becomes a CPN code (in fact, a maximal CPN code) over $X$. By the definition of the operation $\otimes$, it can be also proved that $A \subseteq A \otimes X^n$. However, as the following example shows, there exists a finite CPN code $A$ over $X$ such that there exists no maximal CPN code $B$ over $X$ with $A \subseteq B$. Hence, THEOREM 4.3.2 does not hold for the class of all CPN codes over $X$. □

**EXAMPLE 4.3.3** Consider the language $A = \{ab, aaba, aaa\} \subseteq \{a, b\}^+$. Then this language becomes a CPN code over $\{a, b\}$ (see Fig. 4.1). Moreover, it can be proved that there is no maximal CPN code $B$ over $\{a, b\}$ with $A \subseteq B$ as follows: Suppose $B \subseteq \{a, b\}^+$ is a maximal CPN coce with $A \subseteq B$ over $X$. By LEMMA 4.1.3, $b \in B$ or $b^2 \in B$. Let $b^i \in B$ where $i = 1$ or 2. Let $t_p = p(ab)$ where $p \in P$ and $P$ is the set of places of the Petri net which recognizes $B$. If $p(a) < 0$. Then $p(b) > t_p$ and hence $p(b^i) > t_p$. This contradicts the fact $b^i \in B$. If $p(a) > 0$, then $p(aaba) = p(ab) + 2p(a) > t_p$. This contradicts the fact $aaba \in B$ as well. Hence $p(a) = 0$ and $p(aab) = t_p$. However, since $aab$ is a prefix of $aaba \in B$, $p(aab) < t_p$. This yeilds a contradiction again. Therefore, there is no maximal CPN code $B \subseteq \{a, b\}^+$ with $A \subseteq B$.

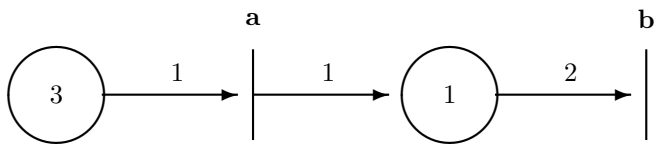



Figure 4.1: Petri net $D$ with $\mathbb{C}(D) = \{ab, aaba, aaa\}$

**REMARK 4.3.4** The set of all maximal CPN codes over $X$ forms a semigroup under $\otimes$. Moreover, the operation $\otimes$ has the following properties:
(1) $A \otimes B = B \otimes A$, (2) $A \otimes A = A$, (3) $A \otimes X = X$.
Consequently, the set of all maximal CPN codes over $X$ forms a commutative band with zero under $\otimes$ (for bands, see [5]).

## 4.4 Rank of CPN Codes

In this section, we will consider the rank and related decomposition of CPN codes.

**DEFINITION 4.4.1** Let $A \subseteq X^+$ be a CPN code over $X$. By $r(A)$ we denote the value $min\{|P| \mid D = (P, X, W, \mu_0), \mathbb{C}(D) = A\}$. □

**REMARK 4.4.1** Let $A \subseteq X^+$ be a finite maximal CPN code over $X$. Then $r(A) \leq |A|$. The proof can be done as follows: Let $D = (P, X, W, \mu_0)$ be a Petri net with a positive initial marking $\mu_0$ such that $\mathbb{C}(D) = A$, and $\delta$ the next-state function of $D$. Let $P' = \{p_u \in P \mid u \in A, p_u(u) = \delta(\mu_0, u)\} \subseteq P$. The transition function $\delta'$ can be defined as $\delta'(\mu|_{P'}, a) = \delta(\mu, a)|_{P'}$ where $a \in X$. Then $A = \mathbb{C}(D')$ and it is obvious that $r(A) \leq |A|$. However, in general this inequality does not hold for a CPN code over $X$ as the following example shows. In Figure 4.2, $\mathbb{C}(D) = \{aba\}$ but $r(\{aba\}) \neq 1$ because $aba \in A$ if and only if $baa \in A$ for any CPN code with $r(A) = 1$. □

Now let $A, B \subseteq X^+$ be maximal CPN codes over $X$. Then it is easy to see that $|A \otimes B| \leq max\{|A|, |B|\}$. Moreover, if $A$ and $B$ are finite, then $r(A \otimes B) \leq r(A) + r(B)$.

We define three language classes as follows: **CPN** $= \{A \subseteq X^+ \mid A$ is a CPN code over $X\}$, **mCPN** $= \{A \subseteq X^+ \mid A$ is an maximal CPN code over $X\}$, **iCPN** $= \{A \subseteq X^+ \mid A$ is a CPN code over $X$, $\exists D = (P, X, W, \mu_0), \forall p \in P, \forall a \in X, W(p, a) \leq 1, \mathbb{C}(D) = A\}$. Then it is obvious that we have the following inclusion relations: **iCPN** $\subseteq$ **mCPN** $\subseteq$ **CPN**. It is also obvious that **mCPN** $\neq$ **CPN**.

**Problem 4.1** *Does* **mCPN** $=$ **iCPN** *hold ?*

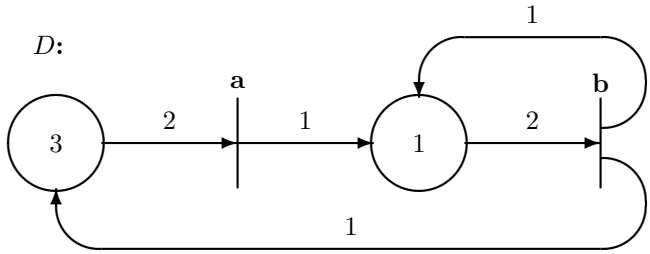



Figure 4.2: Petri net $D$ with $r(\mathcal{L}(D)) > 1$

**PROPOSITION 4.4.1** Let $A \in$ **mCPN**. Then there exist a positive integer $k \geq 1$ and $A_1, A_2, \ldots, A_k \in$ **CPN** such that $r(A_i) = 1, i = 1, 2, \ldots, k$ and $A = A_1 \otimes A_2 \otimes \ldots \otimes A_k$. Moreover, in the above, if $A \in$ **iCPN**, then $A_1, A_2, \ldots, A_k$ are in **iCPN** and context-free.

(Proof) Let $k = r(A)$. Then there exists a Petri net $D = (P, X, W, \mu_0)$ such that $\mathbb{C}(D) = A$ and $|P| = k$. For any $p \in P$, let $D_p = (\{p\}, X, W_p, \{\mu_0(p)\})$ where $W_p$ satisfies $W_p(p, a) = W(p, a)$ and $W_p(a, p) = W(a, p)$ for any $a \in X$. Let $A_p = \mathbb{C}(D_p)$ for an arbitrary $p \in P$. We prove that $A = \otimes_{p \in P} A_p$. Assume $u \in A$. Then there exists $p \in P$ such that $\mu_0(p) = p(u), \mu_0(q) \geq q(u)$ and $\mu_0(p) > q(u')$ for any $q \in P, u' \in P_r(u) \setminus \{u\}$. Therefore, $u \in A_p$ and $(P_r(u) \setminus \{u\}) \cap A_q = \emptyset$ for any $q \in P$. Therefore, $u \in \otimes_{p \in P} A_p$, i.e. $A \subseteq \otimes_{p \in P} A_p$. Now assume $u \in \otimes_{p \in P} A_p$. Then there exists $p \in P$ such that $u \in A_p$ and $(P_r(u) \setminus \{u\}) \cap A_q = \emptyset$ for any $q \in P$. Hence $\mu_0(p) = p(u), \mu_0(q) \geq q(u)$ and $\mu_0(q) > q(u')$ for any $q \in P, u' \in P_r(u) \setminus \{u\}$. Consequently, $u \in A$, i.e. $\otimes_{p \in P} A_p \subseteq A$. Hence $\otimes_{p \in P} A_p = A$. For the proof of the latter half, notice that there exists a Petri net $D = (P, X, W, \mu_0)$ such that $W(p, a) \leq 1$ for all $(p, a) \in P \times X$ and $\mathbb{C}(D) = A$. Let $k = |P|$ where $k \geq r(A)$. Then it is obvious that $A_1, A_2, \ldots, A_k$ are in **iCPN**. Hence it is enough to show that $\mathbb{C}(D)$ is context-free if $D = (\{p\}, X, W, \{n\})$ where $W(p, a) \leq 1$ for any $a \in X$. Now construct the following context-free grammar $G$: $G = (\{S, P\}, X, R, S)$ where $R = \{S \to P^n\} \cup \{P \to a \,|\, a \in X, W(p, a) = 1, W(a, p) = 0\} \cup \{P \to P^{W(a,p)-W(p,a)} \,|\, a \in X, W(a, p) - W(p, a) > 0\}$. Then it can easily be verified that $\mathbb{C}(D) = \mathcal{L}(G)$. □

**Problem 4.2** *In the above proposition, can we take $r(A)$ as $k$ if $A \in$* **iCPN***?*

**PROPOSITION 4.4.2** Let $A \subseteq X^+$ be a finite maximal CPN code with $r(A) = 1$ over $X$. Then $A$ is a full uniform code over $X$.

(Proof) Let $D = (\{p\}, X, W, \{n\})$ be a Petri net with $\mathbb{C}(D) = A$ and let $u, v \in A$. Then $p(u) = p(v) = n$. By LEMMA 4.1.1, $A$ is a full uniform code over $X$. □

**PROPOSITION 4.4.3** Let $A \subseteq X^+$ be a maximal CPN code with $r(A) = 1$ over $X$ and let $k$ be a positive integer. Then $A^k$ is a maximal CPN code with $r(A^k) = 1$ over $X$.

(Proof) Let $D = (\{p\}, X, W, \{n\})$ be a Petri net with $\mathbb{C}(D) = A$. Now let $D_k = (\{p\}, X, W, \{kn\})$. Then it can easily be seen that $A^k$ is a maximal prefix code over $X$ and $\mathbb{C}(D_k) = A^k$. Hence $A^k$ is a maximal CPN code over $X$ with $r(A^k) = 1$. □

**PROPOSITION 4.4.4** Let $A \in$ **iCPN**. Then, by PROPOSITION 4.4.3, there exist $A_1, A_2, \ldots, A_k \in$ **iCPN** such that $r(A_i) = 1, i = 1, 2, \ldots, k$ and $A = A_1 \otimes A_2 \otimes \ldots \otimes A_k$. Let $n_1, n_2, \ldots, n_k$ be positive integers. Then $A_1^{n_1} \otimes A_2^{n_2} \otimes \ldots, \otimes A_k^{n_k} \in$ **iCPN**. □

(Proof) Obvious from the above results. □

## 4.5 Context-sensitiveness of CPN Codes

Consider the Petri net $D = (P, X, W, \mu_0)$ depicted below. Then $\mathbb{C}(D) \cap a^+b^+c^+ = \cup_{n \geq 1}\{a^n b^i c^{n+i+1} | 1 \leq i \leq n\}$ is not context-free. Hence $\mathbb{C}(D)$ is not context-free. Therefore, the class of all CPN codes over an alphabet $X$ is not necessary included in the class of all context-free languages over $X$. However, in this section, we will prove the context-sensitiveness of CPN languages in two different ways, i.e. the first one is an indirect proof and the second one is a direct proof.

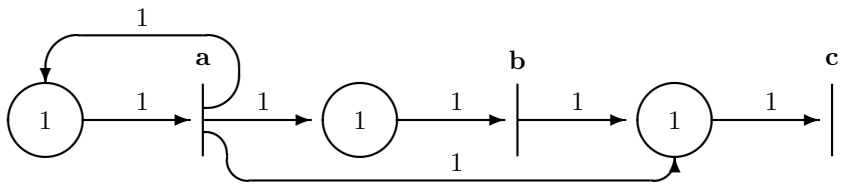



Figure 4.3: Petri net which generates a non-context-free language

**THEOREM 4.5.1** Let $C \subseteq X^+$ be a CPN code over $X$. Then $C$ is a context-sensitive language over $X$.

For the first proof, we will use the following results: 1) The complement of a context-sensitive language is context-sensitive ([43]). 2) The concatenation of two context-sensitive languages is context-sensitive. 3) A Petri net language of type $G$ is context-sensitive ([23]).

(First proof) Let $\mu_1, \mu_2$ be markings of a Petri net $D = (P, X, W)$. Then, $\mu_1 \leq \mu_2$ means that $\mu_1(p) \leq \mu_2(p)$ for any $p \in P$. Now let $D = (P, X, W, \mu_0)$ be a Petri net with an initial marking $\mu_0 \in N^P$ such that $C = \mathbb{C}(D)$. Let $F = \{\mu \,|\, \Sigma_{p \in P}\mu(p) < |P|\}$. Moreover, Let $L_1 = \{w \in X^* \,|\, \exists \mu \in F, \delta(\mu_0, w) \geq \mu\}$, i.e. $L_1$ is a Petri net language of type $G$. Therefore, $L_1$ is context-sensitive. Now let $L_2 = \{w \in X^* \,|\, \forall p \in P, \delta(\mu_0, w)(p) \geq 1\}$. Then $L_2$ is also context-sensitive. It is obvious that $L_1 \setminus L_2 = \{w \in X^* \,|\, \exists p \in P, \delta(\mu_0, w)(p) = 0\}$ and this language is context-sensitive. Consider $(L_1 \setminus L_2) \setminus (L_1 \setminus L_2)X^+$. Then $(L_1 \setminus L_2) \setminus (L_1 \setminus L_2)X^+ = \{w \in X^* \,|\, \exists p \in P, \delta(\mu_0, w)(p) = 0$ and $\forall p \in P, \delta(\mu_0, w')(p) > 0$ for $w' \in P_r(w) \setminus \{w\}\} = \mathbb{C}(D)$. Hence $C$ is context-sensitive. □

Before giving the second proof, we provide a few notations. Let $\mu_1, \mu_2, \ldots, \mu_r$ and $\mu$ be markings of a Petri net. Then $\mu = \mu_1 + \mu_2 + \ldots + \mu_r$ if $\mu(p) = \mu_1(p) + \mu_2(p) + \ldots + \mu_r(p)$ for any $p \in P$. Now let $D = (P, X, W, \mu_0)$ be a Petri net with a positive initial marking $\mu_0$. Let $N_D = max\{W(p, a), W(b, q) \,|\, a, b \in X, p, q \in P\}$ and let $M_D = max\{\mu_0(p) \,|\, p \in P\}$. By $\Omega_D$ we denote the set of markings $\{\mu \,|\, \forall p \in P, \mu(p) \leq M_D + 3N_D\}$. Notice that $\Omega_D$ is a finite set.

(Second proof) Let $D = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$. $\delta$ is the next-state function of $D$. We construct the following context-sensitive grammar $G = (V, X, R, S)$ where $V$ is the set of variables, $X$ is an alphabet, $R$ is a set of productions (rewriting rules) and $S$ is a start symbol, as follows: $V = \{S, [\delta]\} \cup \{[w] \,|\, w \in X^2 \cup X^3\} \cup \{[\mu] \,|\, \mu \in \Omega_D\} \cup \{[\pi_p] \,|\, p \in P\}$ and $R = R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5 \cup R_6 \cup R_7 \cup R_8$, where

$$
\begin{aligned}
R_1 &= \{S \to w \,|\, w \in (X \cup X^2 \cup X^3) \cap \mathbb{C}(D)\},\\
R_2 &= \{S \to [\delta][\mu_0]\},\\
R_3 &= \{[\delta][\mu] \to [w][\delta][\nu][\nu'] \,|\, \mu \in \boldsymbol{N}^P \cap \Omega_D, w \in X^2 \cup X^3,\\
&\qquad \nu + \nu' = \delta(\mu, w), \nu, \nu' \in \Omega_D, \forall w' \in P_r(w), \delta(\mu, w') \in \boldsymbol{N}^P\},\\
R_4 &= \{[\mu][\nu] \to [\mu'][\nu'] \,|\, \mu + \nu = \mu' + \nu', \mu, \nu, \mu', \nu' \in \Omega_D\},\\
R_5 &= \{[\delta][\mu] \to [w][\pi_p] \,|\, p \in P, \mu \in \boldsymbol{N}^P \cap \Omega_D, w \in X^2 \cup X^3,\\
&\qquad \forall w' \in P_r(w) \setminus \{w\}, \delta(\mu, w') \in \boldsymbol{N}^P, \delta(\mu, w)(p) = 0\},\\
R_6 &= \{[\pi_p][\mu] \to [\pi_p][\pi_p] \,|\, p \in P, \mu \in \Omega_D, \mu(p) = 0\},\\
R_7 &= \{[w][\pi_p] \to [\pi_p][w] \,|\, p \in P, w \in X^2 \cup X^3\},\\
R_8 &= \{[w][\pi_p] \to w \,|\, p \in P, w \in X^2 \cup X^3\}.
\end{aligned}
$$

We provide the following lemma.

**LEMMA 4.5.1** Let $k$ be a positive integer. Then $S \Rightarrow^* [w_1][w_2] \ldots [w_k][\delta][\mu_1][\mu_2] \ldots [\mu_k][\mu_{k+1}]$ if and only if $\delta(\mu_0, w_1 w_2 \ldots w_k) = \mu_1 + \mu_2 + \ldots + \mu_k + \mu_{k+1}$ where $\mu_i \in \Omega_D, i = 1, 2, \ldots k+1, w_j \in X^2 \cup X^3, j = 1, 2, \ldots, k$ and $\delta(\mu_0, w) \in \boldsymbol{N}^P$ for any $w \in P_r(w_1 w_2 \ldots w_k)$.

(Proof of Lemma 5.1) Consider the case $k = 1$. Then $S \Rightarrow^* [w_1][\delta][\mu_1][\mu_2]$ implies that $S \Rightarrow [\delta][\mu_0] \Rightarrow [w_1][\delta][\mu_1][\mu_2]$ where $\delta(\mu_0, w_1) = \mu_1 + \mu_2$. Moreover, we have $\delta(\mu_0, w) \in \boldsymbol{N}^P$ for any $w \in P_r(w_1)$. Now assume that $\delta(\mu_0, w_1) = \mu_1 + \mu_2$ and $\delta(\mu_0, w) \in \boldsymbol{N}^P$ for any $w \in P_r(w_1)$. Then, by $R_2$ and $R_3$, we have

$S \Rightarrow^* [w_1][\delta][\mu_1][\mu_2]$. Thus the lemma holds for $k = 1$. Assume that the lemma holds for $k = 1, 2, \ldots, n$. Let

$$S \Rightarrow^* [w_1][w_2]\ldots[w_n][w_{n+1}][\delta][\mu_1][\mu_2]\ldots[\mu_{n+1}][\mu_{n+2}].$$

Then we have the following derivation:

$S \Rightarrow^* [w_1][w_2]\ldots[w_n][\delta][\nu_1][\nu_2]\ldots[\nu_n][\nu_{n+1}] \Rightarrow [w_1][w_2]\ldots[w_n][w_{n+1}]$
$[\delta][\nu_1'][\nu_2''][\nu_2]\ldots[\nu_n][\nu_{n+1}] \Rightarrow^* [w_1][w_2]\ldots[w_n][w_{n+1}][\delta][\mu_1][\mu_2]\ldots[\mu_{n+1}]\ [\mu_{n+2}]$

In the above, $\delta(\nu_1, w_{n+1}) = \nu_1' + \nu_1''$ and $\delta(\nu_1, w) \in N_+^P$ for any $w \in P_r(w_{n+1})$. Moreover, $\nu_1' + \nu_1'' + \nu_2 + \ldots + \nu_{n+1} = \mu_1 + \mu_2 + \ldots + \mu_{n+1} + \mu_{n+2}$. On the other hand, by assumption, $\delta(\mu_0, w_1w_2\ldots w_n) = \nu_1 + \nu_2 + \ldots + \nu_n + \nu_{n+1}$ and $\delta(\mu_0, w) \in N_+^P$ for any $w \in P_r(w_1w_2\ldots w_n)$. Since $\delta(\mu_0, w_1w_2\ldots w_n) = \nu_1 + \nu_2 + \ldots + \nu_n + \nu_{n+1}$ and $\delta(\nu_1, w_{n+1}) = \nu_1' + \nu_2''$, $\delta(\mu_0, w_1w_2\ldots w_nw_{n+1}) = \delta(\nu_1, w_{n+1}) + \nu_2 + \ldots + \nu_n + \nu_{n+1} = \nu_1' + \nu_2'' + \nu_2 + \ldots + \nu_n + \nu_{n+1} = \mu_1 + \mu_2 + \ldots + \mu_{n+1} + \mu_{n+2}$. Moreover, since $[\delta][\nu_1] \rightarrow [w_{n+1}][\delta][\nu_1'][\nu_2'']$ is applied, $\delta(\nu_1, w') \in N_+^P$ for any $w' \in P_r(w_{n+1}) \setminus \{w_{n+1}\}$. Remark that $\delta(\mu_0, w_1w_2\ldots w_n) = \nu_1 + \nu_2 + \ldots + \nu_n + \nu_{n+1}$. Therefore, $\delta(\mu_0, w_1w_2\ldots w_nw') = \delta(\nu_1, w') + \nu_2 + \ldots + \nu_n + \nu_{n+1} \in \boldsymbol{N}^P$ for any $w_1w_2\ldots w_nw' \in P_r(w_1w_2\ldots w_nw_{n+1})$. Together with the previous assumption, we have $\delta(\mu_0, w) \in \boldsymbol{N}^P$ for any $w \in P_r(w_1w_2\ldots w_nw_{n+1})$. Now assume that $\delta(\mu_0, w_1w_2\ldots w_nw_{n+1}) = \mu_1 + \mu_2 + \ldots + \mu_{n+1} + \mu_{n+2}$ where $\mu_i \in \Omega_D, i = 1, 2, \ldots, n+2$ and $\delta(\mu_0, w) \in \boldsymbol{N}_+^P$ for any $w \in P_r(w_1w_2\ldots w_nw_{n+1})$. Consider $\delta(\mu_0, w_1w_2\ldots w_n) \in \boldsymbol{N}^P$. Since $\pi_p(\delta(\mu_0, w_1w_2\ldots w_n)) \leq N_D + 3nM_D \leq n(N_D + 3M_D)$, $\delta(\mu_0, w_1w_2\ldots w_n)$ can be represented as $\nu_1 + \nu_2 + \ldots + \nu_n + \nu_{n+1}$ where $\nu_1, \nu_2, \ldots, \nu_n, \nu_{n+1} \in \Omega_D$. On the other hand, $\delta(\mu_0, w) \in N_+^P$ for any $w \in P_r(w_1w_2\ldots w_n)$. By induction hypothesis, $S \Rightarrow^* [w_1][w_2]\ldots[w_n][\delta][\nu_1][\nu_2]$
$\ldots[\nu_{n+1}]$ where $\delta(\mu_0, w_1w_2\ldots w_n) = \nu_1 + \nu_2 + \ldots + \nu_{n+1}$. Note that $\delta(\mu_0, w_1w_2\ldots w_nw_{n+1})$ can be computed and $\delta(\mu_0, w_1w_2\ldots w_nw') \in \boldsymbol{N}^P$ for any $w' \in P_r(w_{n+1})$. Moreover, since $max\{|\pi_p(\delta(\mu_0, w_1w_2\ldots w_nu))| - |\pi_p(\delta(\mu_0, w_1w_2\ldots w_n))| \mid p \in P, |u| \leq 3\} \leq 3M_D$, There exist $\gamma_1, \gamma_2, \ldots, \gamma_{n+1} \in \Omega_D$ such that $\nu_1 + \nu_2 + \ldots + \nu_{n+1} = \gamma_1 + \gamma_2 + \ldots + \gamma_{n+1}$ and $\delta(\gamma_1, w') \in \boldsymbol{N}^P$ for any $w' \in P_r(w_n)$. Therefore, we have $S \Rightarrow^* [w_1][w_2]\ldots[w_n][\delta][\nu_1][\nu_2]\ldots[\nu_{n+1}] \rightarrow [w_1][w_2]\ldots[w_n][\delta][\gamma_1][\gamma_2]\ldots[\gamma_{n+1}]$. Notice that the rule $[\delta][\gamma_1] \rightarrow [w_1][\delta][\gamma_1'][\gamma_1'']$ can be applied. Hence we have $S \Rightarrow^* [w_1][w_2]\ldots[w_n][w_{n+1}][\delta][\gamma_1'][\gamma_1''][\gamma_2]\ldots[\gamma_{n+1}]$. Since $\delta(\mu_0, w_1w_2\ldots w_nw_{n+1}) = \mu_1 + \mu_2 + \ldots + \mu_{n+1} + \mu_{n+2} = \gamma_1' + \gamma_1'' + \gamma_2 + \ldots + \gamma_{n+1}$, $S \Rightarrow^* [w_1][w_2]\ldots[w_n][w_{n+1}][\delta][\gamma_1'][\gamma_1''][\gamma_2]\ldots[\gamma_{n+1}] \Rightarrow^* [w_1][w_2]\ldots[w_n][w_{n+1}]$
$[\delta][\mu_1][\mu_2]\ldots[\mu_{n+1}][\mu_{n+2}]$. This completes the proof of the lemma. □

Now we return to the second proof. Let $u \in \mathbb{C}(D)$. If $|u| \leq 3$, then $S \Rightarrow u$ and $u \in \mathbb{C}(G)$. Assume $|u| > 3$. Then $u = w_1w_2\ldots w_n$ for some $w_i \in X^2 \cup X^3, i = 1, 2, \ldots, n$. Notice that $|p(w_i)| \leq 3M_D$ for any $p \in P, w_i, i = 1, 2, \ldots, n-1$. Hence there exist $\mu_1, \mu_2, \ldots, \mu_n \in \Omega_D$ such that $\delta(\mu_0, w_1w_2\ldots w_{n-1}) = \mu_1 + \mu_2 + \ldots + \mu_n$. Moreover, it is obvious that $\delta(\mu_0, w) \in \boldsymbol{N}^P$ for any $w \in P_r(w_1w_2\ldots w_{n-1})$. By LEMMA 4.5.1, $S \Rightarrow^* [w_1][w_2]\ldots[w_{n-1}][\delta][\mu_1][\mu_2]\ldots[\mu_n]$. Let $\nu_1, \nu_2, \ldots, \nu_n \in \Omega_D$ such that $\nu_1(p) = min\{\pi_p(\mu_1 + \mu_2 + \ldots + \mu_n), N_D + 3M_D\}$ for any $p \in P$ and $\nu_1 + \nu_2 + \ldots + \nu_n = \mu_1 + \mu_2 + \ldots + \mu_n$. Then $S \Rightarrow^*$

$[w_1][w_2]\dots[w_{n-1}][\delta][\mu_1][\mu_2]\dots[\mu_n] \Rightarrow [w_1][w_2]\dots[w_{n-1}][\delta][\nu_1][\nu_2]\dots[\nu_n]$. By the definition of $\nu_1$ and $u = w_1w_2\dots w_{n-1}w_n \in \mathbb{C}(D)$, $\delta(\mu_0, w_1w_2\dots\ w_{n-1}w) \in N_+^P$ for any $p \in P$ and $w \in P_r(w)\setminus\{w\}$, and $(\delta(\mu_0, w_1w_2\dots w_{n-1}w_n)(q) = 0$ for some $q \in P$. Thus the rule $[\delta][\nu_1] \to [w_n][\pi_q]$ can be applied and we have $S \Rightarrow^* [w_1][w_2]\dots[w_{n-1}][\pi_q][\nu_2]\dots[\nu_n]$. Since $\pi_q(\nu_2+\nu_3+\dots+\nu_n) = 0, \pi_q(\nu_i) = 0$ for any $i, i = 2, 3, \dots, n$. Thus we have $S \Rightarrow^* [w_1][w_2]\dots[w_{n-1}][w_n][\pi_q]^n \Rightarrow^* [w_1][\pi_q][w_2][\pi_q]\dots[w_{n-1}][\pi_q][w_n][\pi_q] \Rightarrow^* w_1w_2\dots w_n$. Consequently, $u = w_1w_2\dots w_n \in \mathcal{L}(G)$ and $\mathbb{C}(D) \subseteq \mathcal{L}(G)$.

Let $u = w_1w_2\dots w_{n-1}w_n \in \mathcal{L}(G)$. Then we have the following derivation:

$S \Rightarrow^* [w_1][w_2]\dots[w_{n-1}][\delta][\mu_1][\mu_2]\dots[\mu_{n-1}][\mu_n] \Rightarrow [w_1][w_2]\dots[w_{n-1}][w_n][\pi_p][\mu_2]\dots[\mu_{n-1}][\mu_n] \Rightarrow^* [w_1][w_2]\dots[w_{n-1}][w_n][\pi_p]^n \Rightarrow^* [w_1][\pi_p][w_2][\pi_p]\dots[w_{n-1}][\pi_p][w_n][\pi_p] \Rightarrow^* w_1w_2\dots w_{n-1}w_n$.

By LEMMA 4.5.1, $\delta(\mu_0, w) \in \boldsymbol{N}^P$ for any $w \in P_r(w_1w_2\dots w_{n-1})$. Since the rules $[\delta][\mu_1] \to [w_n][\pi_p], [\pi_p][\mu_i] \to [\pi_p][\pi_p], i = 2, 3, \dots, n$ are applied, $\delta(\mu_0, w) \in \boldsymbol{N}^P$ for any $w \in P_r(w_1w_2\dots w_{n-1}w_n) \setminus \{w_1w_2\dots w_{n-1}w_n\}$ and $\delta(\mu_0, w_1w_2\dots\ w_{n-1}w_n)(p) = 0$. This means that $u = w_1w_2\dots w_{n-1}w_n \in \mathbb{C}(D)$, i.e. $\mathcal{L}(G) \subseteq \mathbb{C}(D)$. Consequently, $\mathcal{L}(G) = \mathbb{C}(D)$. □

## 4.6 Remarks and Further Works

In Section 4.5, we show that the family **CPNC** of all CPN codes is included in $CS$ but not in $CF$ by both an indirect proof and a direct proof. By the definitions, it is immediate that the family **DPN** of all DPN codes is a subset of the family **SPN** of all SPN codes, and the family **BPN** of all BPN codes is a subset of the family **CPN** of all CPN codes. But it has not been investigated which language class each family of **DPN**, **SPN**, **BPN** is included in exactly.

The derivation (computation) by context-sensitive grammar $G$ in the second poof of THEOREM 4.5.1 simulates the behavior of a general Petri net and it can halt when a final condition is satisfied. This context-sensitive framework is effective on recognizing Petri net language and codes of other classes. I think that this framework is applicable to directly show the context-sensitiveness of a language generated by a mechanical system as well as a Petri net.

# Chapter 5

# Maximality of CPN Codes

In Chapter 5 we treat the open question raised in Chapter 4. A CPN code generated by some input-ordinary Petri net is called an input-ordinary CPN code (iCPNC, for short) and obviously a maximal CPN code. The problem is whether **mCPNC** = **iCPNC** or not, where **mCPNC**(resp., **iCPNC**) means the family of maximal CPN codes (resp., input-ordinary CPN codes). It is easily seen that the latter is a subfamily of the former. But the reverse inclusion is still open in a general Petri net. So we show that the inclusion is true in restricted cases, i.e., the case that the number of places is $\leq 2$, and the case that the number of transitions is equal to 1.

## 5.1 Fundamental Properties

Here we state some fundamental properties used in the following sections.

**DEFINITION 5.1.1** Let $PN = (P, X, W, \mu_0)$ be a Petri net and $\mu_0$ be a positive marking. For $w \in X^*$ and $a \in X$, the set $K_w(a)$ of places is defined as follows. If $\delta(\mu_0, w)$ is not defined, then $K_w(a) = \emptyset$. Otherwise,

$$K_w(a) = \{p \in P \mid \delta(\mu_0, w) = \mu,\ W(a,p) = 0, \exists n \in \boldsymbol{N},$$
$$(\mu + n \cdot \Delta(a))(p) = 0, \quad \text{and} \quad (\mu + n \cdot \Delta(a))(q) \geq 0 \text{ for } \forall q \in P \setminus \{p\})\}.$$

An element of $K_w(a)$ is called a critical place (after reading the word $w$). Especially $K_w(a)$ is denoted by $K(a)$ when $w = 1$(the empty word). $K_w$ is a mapping from $X$ to $2^P$, called the critical place mapping of the Petri net $PN$. □

A critical place $p$ of a transition $a$ means that $p$ is a place where the number of tokens first becomes zero when $a$ fires one after another(see Figure 5.1).

**THEOREM 5.1.1 (Fundamental Theorem)** Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$, $K$ be its critical place mapping. If

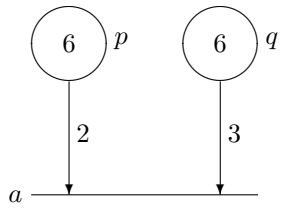



Figure 5.1: Example of a critical place, $K(a) = \{q\}$.

$C = \mathbb{C}(PN)$ is a maximal CPN code, then for any $p \in P$ and $a, b \in X$ the following conditions hold.

(1) $p \in K(a)$ implies $W(p,a) \geq W(p,b)$,
(2) $p \in K(a) \cap K(b)$ implies $W(p,a) = W(p,b)$.

□

(Proof) (1) Since $p \in K(a)$, there exists some nonnegative integer $n$ such that $a^{n+1} \in C$ and $\delta(\mu_0, a^n) = W(p,a)$. Then by the maximality of $C$ each transition $b \in X$ must be enable. Therefore $W(p,a) \geq W(p,b)$.

(2) Since $W(p,a) \geq W(p,b)$ and $W(p,b) \geq W(p,a)$ hold by (1), we have the equality $W(p,a) = W(p,b)$. □

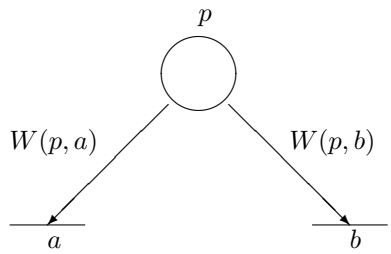



Figure 5.2: $p \in K(a) \Rightarrow W(p,a) \geq W(p,b)$.

**THEOREM 5.1.2 (Deletion of useless places)** Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$, $C = \mathbb{C}(PN)$ be a maximal prefix code. Let $p \in P$ be a place such that $\delta(\mu_0, w)(p) \neq 0$ for any $w \in C$. And the Petri net $PN' = (P', X, W', \mu_0')$ is defined as follows:

$P' = P \setminus \{p\}$,
$W'$ is a restriction of $W$ on $(P' \times X) \cup (X \times P')$,
$\mu_0'$ is a restriction of $\mu_0$ on $P'$.

Then,

$$\mathbb{C}(PN) = \mathbb{C}(PN').$$

$PN'$ is obtained from $PN$ by deleting the place $p$ and its all input/output arcs attached to $p$. □

(Proof) Let $\delta$ and $\delta'$ be the next-state functions of the Petri nets $PN$ and $PN'$ respectively. We will prove by induction. At first $\mu'_0 = \mu_0|P'$ holds. Next let $u$ be an arbitrary word with $uX^+ \cap C \neq \emptyset$. Suppose that $\delta'(\mu_0, u) = \delta(\mu_0, u)|P'$ holds. By the maximality of $C$, $\delta(\mu_0, u)(q) \geq W(q, a)$ for any $a \in X$ and $q \in P$. This implies that $a \in X$ is enable both $PN$ and $PN'$ after reading the firing sequence $u$. Since

$$\delta'(\mu'_0, ua) = \delta'(\delta'(\mu'_0, u), a) = \delta'(\delta(\mu_0, u)|P', a) = \delta(\delta(\mu_0, u), a)|P' = \delta(\mu_0, ua)|P'$$

and $\delta(\mu_0, w)(p) \neq 0$ for any $w \in C$, $\delta'(\mu'_0, ua)$ is positive iff $\delta(\mu_0, ua)$ is positive. Therefore $\mathbb{C}(PN) = \mathbb{C}(PN')$. □

Generally set $P_0 = \{q \in P \mid \exists w \in C, \delta(\mu_0, w)(q) = 0\}$. Applying the above theorem repetitively, the theorem holds even if we replace $P'$ in the theorem with $P_0$. The maximality in the theorem is needed because the following counter example exists.

**EXAMPLE 5.1.1** Let $P = \{p, q\}$, $X = \{a, b\}$, $W(p, a) = W(p, b) = 1, W(q, b) = 2$, $\mu_0(p) = \mu_0(q) = 1$. The weights of any other arcs are all zero. Then $C = \mathbb{C}(P, X, W, \mu_0) = \{a\}$ is not a maximal CPN code. For any $w \in C$, $\delta(\mu_0, w)(q) \neq 0$, where $\delta$ is the next-state function of $(P, X, W, \mu_0)$. However, Since $P' = P \setminus \{q\} = \{p\}$, $W'(p, a) = W'(p, b) = 1$, $\mu'_0(p) = 1$, The weights of any other arcs are all zero, $C' = \mathbb{C}(P', X, W', \mu'_0) = \{a, b\}$. This shows that $C = C'$ does not necessarily hold if $C$ is not a maximal CPN code. □

**THEOREM 5.1.3 (Reduction rule of two-way arcs)** Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$. Let $C = \mathbb{C}(PN)$ be a maximal prefix code. Let $p \in P$, $a \in X$ with $W(p, a) > 0$ and $W(a, p) > 0$. Then the Petri net $PN' = (P, X, W', \mu_0)$ is defined as follows, which is obtained by replacing the weights of the two arcs $(p, a)$ and $(a, p)$.

$$\begin{array}{ll}
W(p,a) > W(a,p) & \Rightarrow W'(p,a) = W(p,a) - W(a,p),\ W'(a,p) = 0 \\
W(p,a) = W(a,p) & \Rightarrow W'(p,a) = W'(a,p) = 0 \\
W(p,a) < W(a,p) & \Rightarrow W'(a,p) = W(a,p) - W(p,a),\ W'(p,a) = 0 \\
q \neq p \text{ or } b \neq a & \Rightarrow W'(b,q) = W(b,q), W'(q,b) = W(q,b)
\end{array}$$

Then

$$\mathbb{C}(PN) = \mathbb{C}(PN').$$

□

(Proof) Let $\delta$ and $\delta'$ be the next-state functions of the Petri nets $PN$ and $PN'$ respectively. Suppose that $\delta'(\mu_0, u) = \delta(\mu_0, u)$ for any $u \in X^*$ with

$uX^+ \cap C \neq \emptyset$. Then for any $b \in X$ we show that $\delta'(\mu_0, ub) = \delta(\mu_0, ub)$. Since $\mathbb{C}(PN)$ is a maximal prefix code, $\delta(\mu_0, u)(q) \geq W(q, b)$ for any $q \in P$ holds.

(i) In case of $b \neq a$. $b$ is clearly enable in $PN'$, and $\delta'(\mu_0, ub)(q) = \delta(\mu_0, ub)(q)$ holds.

(ii) In case of $b = a$. For any $q \in P$, $\delta'(\mu_0, u)(q) = \delta(\mu_0, u)(q) \geq W(q, a) \geq W'(q, a)$. So $a$ is enable after reading $u$ in both $PN$ and $PN'$. For any $q \in P$,

$$\begin{aligned} \delta'(\mu_0,\, ua)(q) &= \delta'(\mu_0,\, u)(q) - W'(q, a) + W'(a, q) \\ &= \delta(\mu_0,\, u)(q) - W(q, a) + W(a, q) \\ &= \delta(\mu_0,\, ua)(q). \end{aligned}$$

By (i) and (ii), for arbitrary $u$ with $uX^* \cap C \neq \emptyset$, $\delta'(\mu'_0, u)$ is positive iff $\delta(\mu_0, u)$ is positive. Therefore $\mathbb{C}(PN) = \mathbb{C}(PN')$. □

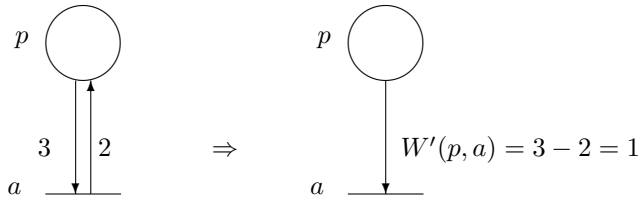



Figure 5.3: Replace the weights of an arc with their difference.

**EXAMPLE 5.1.2** Let $X$ be an alphabet and $k$ be a positive integer. Suppose that subsets $X_1$ and $X_2$ of $X$ satisfy $X = X_1 \cup X_2$ and $X_1 \cap X_2 = \emptyset$. Then, the following language $C$ is an input-ordinary CPN code.

$$C = (\bigcup_{0 \leq i < k} {X_2}^i X_1) \cup {X_2}^k.$$

□

(proof) Let $PN = (P, X, W, \mu_0)$ be the input-ordinary Petri defined as follows (see Figure 5.4):

$$\begin{array}{lll} P = \{p, q\},\ X = X_1 \cup X_2, & & \\ \mu_0(p) = 1, \quad \mu_0(q) = k, & & \\ W(p, a) = 1, W(a, p) = 0 & \text{if} & a \in X_1 \\ W(q, b) = 1, W(b, q) = 0 & \text{if} & b \in X_2 \end{array}$$

Then $C = \mathbb{C}(PN)$ clearly holds. □

Especially, in this example by setting $X_1 = \emptyset$ and $X_2 = X$, then $C = X^k = \{w \in X^* \,|\, |w| = k\}$. $X^k$ is called a (full) uniform code over $X$. Therefore a uniform code becomes an input-ordinary CPN code.

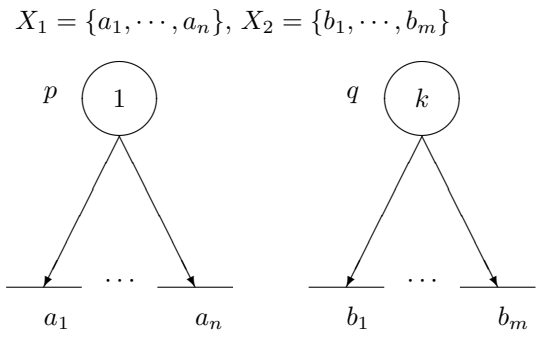



Figure 5.4: the Petri net generating $C$ of Example .

### 5.1.1 In the case $|P| = 1$ or $|X| = 1$

At first we consider the case the number $|P|$ of places equals 1 and the case the number $|X|$ of transitions equals 1.

**THEOREM 5.1.4** Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$. Assume that $|X| = 1$ or $|P| = 1$. If $C = \mathbb{C}(PN)$ is a maximal prefix code, then $C$ is an input-ordinary CPN code. □

(Proof) In case of $|X| = 1$. Let $X = \{a\}$. Since $C$ is a maximal prefix code, there exists some positive integer $n$ with $C = \{a^n\}$, that is, $C$ is a uniform code. As we see in EXAMPLE 5.1.2, $C$ is an input-ordinary CPN code.

In case of $|P| = 1$, let $P = \{p\}$. Since $C$ is a maximal prefix code, there exists $b \in X$ such that $w_0 = W(p, b) > 0$ and $W(b, p) = 0$. Then $w_0$ is a divisor of $\mu_0(p)$. Moreover, for any $\forall a \in X \setminus \{b\}$ $W(p, a)$ must be equal to either 0 or $w_0$, and $W(a, p)$ must be equal to either 0 or a multiple of $w_0$. Therefore $C$ is an input-ordinary CPN code. □

Assume that $|P| = 1$, that is $P = \{p\}$ in this theorem. Setting $X_1 = \{a \in X | W(p, a) > 0, W(a, p) = 0\}$ and $X_2 = X - X_1$, Then

$$C(P, X, W, \mu_0) = ({X_1}^{n-1} \diamond (\bigcup_{a_i \in X_2} a_i {X_1}^{n_i})^{\diamond}) X_1,$$

where $n_i = W(a_i, p)/n$, $\diamond$ is the shuffle product over two languages $L$, $K \subset X^*$ defined by $L \diamond K = \cup_{x \in L,\, y \in K} x \diamond y$, $x \diamond y = \{x_1 y_1 x_2 y_2 \cdots x_n y_n | \, x = x_1 x_2 \cdots x_n,\ y = y_1 y_2 \cdots y_n,\ x_i, y_i \in X^* \text{ for } 1 \leq i \leq n\}$ for $x, y \in X^*$ and $L^{\diamond}$ is the shuffle closure of a language $L$, defined by $L^{\diamond} = \cup_{i \geq 0} L^{\diamond i}$, $L^{\diamond 0} = \{1\}$, $L^{\diamond(i+1)} = L^{\diamond i} \diamond L$.

In case that a Petri net has only a place or only a transition, we have proved that **mCPNC=iCPNC**. In the following section, we consider the case that a Petri net has two places.

## 5.2 Maximal CPN Codes with two Places

Here we solve the problem whether **mCPNC**⊆**iCPNC** holds or not under the conditions that a Petri net has just two places. All through this section, we assume that a Petri net $PN = (P, X, W, \mu_0)$ with a positive marking $\mu_0$ generating a code satisfies the following conditions without the loss of generality.
(1) $|P| = 2$, Set $P = \{p, q\}$.
(2) $X \neq \emptyset$ and $X$ includes no isolated transition because a marking is unchanged by a isolated transition's firing.

Moreover if $\mathbb{C}(PN)$ is a maximal CPN code, we implicitly assume that $PN$ satifies the next useful conditions.
(3) Every arc is one way by THEOREM 5.1.3. That is, for any $p \in P$ and $a \in X$, $W(a, p) = 0$ or $W(p, a) = 0$ .
(4) $PN$ has no useless place by THEOREM 5.1.2. That is, for any $p \in P$, there exists $w \in \mathbb{C}(PN)$ with $\delta(\mu_0, w)(p) = 0$.

### 5.2.1 Without Source Transitions

In this subsection, each transition in $X$ is either a sink transition or a transform transition.

**THEOREM 5.2.1** Let $PN = (P, X, W, \mu_0)$ be a Petri net without source transitions, $\mu_0$ be positive and $P = \{p, q\}$. If $C = \mathbb{C}(PN)$ is a maximal CPN code, then $C$ is an input-ordinary CPN code.

(proof) Setting $X_p$ and $X_q$ as follows:

$$X_p = \{a \in X \mid p \in K(a)\} = K^{-1}(\{p\}),$$
$$X_q = \{a \in X \mid q \in K(a)\} = K^{-1}(\{q\}).$$

(note that $X_p \cap X_q = K^{-1}(\{p, q\}) = \emptyset$ does not necessarily hold), where $K$ is the critical place mapping of $PN$.

Since an arbitrary transition $a \in X$ is a sink or transform transition by the condition (3), the number of tokens in $p$ or $q$ becomes zero when $a$ fires in succession, that is, $X = X_p \cup X_q$ holds.

By THEOREMs 5.1.1 and 5.1.3 there exist some positive integers $n_p$ and $k$ such that $W(p, a) = n_p$, $W(a, p) = 0$ and $\mu_0(p) = kn_p$ for any $a \in X_p$. Similarly, there exist some positive integers $n_q$ and $l$ such that $W(q, a) = n_q$, $W(a, q) = 0$ and $\mu_0(q) = ln_q$ for any $a \in X_q$

If $k = l = 1$, the statement of this theorem holds because the code $C$ is the uniform code $X^1$. If $X_p = \emptyset$, that is $X = X_q$, then $C$ is the uniform code $C = X^l \in$ **iCPNC**. Similarly $C$ is also a uniform code $C = X^k$ if $X_q = \emptyset$. So we may assume that $k \cdot l > 1$, $X_p \neq \emptyset$ and $X_q \neq \emptyset$ hold.

If there exists neither $a \in X_p$ such that $n_q \nmid W(q, a)$ or $n_q \nmid W(a, q)$, nor $b \in X_q$ such that $n_p \nmid W(p, b)$ or $n_p \nmid W(a, b)$, then the weight of each output arc from the place $p$(resp., $q$) is zero or $n_p$(resp., $n_q$), the weight of every input arc to $p$(resp., $q$) is a multiple of $n_p$(resp., $n_q$). Therefore, $\mathbb{C}(PN)$

is the same as $\mathbb{C}(PN')$ generated by the following input-ordinary Petri net $PN' = (P', X', W', \mu_0')$:

$P' = P = \{p, q\}$, $X' = X$,
$W'(p, a) = W(p, a)/n_p \in \{0, 1\}$, $W'(a, p) = W(a, p)/n_p \in \boldsymbol{N}_0$ for $\forall a \in X$,
$W'(q, a) = W(q, a)/n_q \in \{0, 1\}$, $W'(a, q) = W(a, q)/n_q \in \boldsymbol{N}_0$ for $\forall a \in X$,
$\mu_0'(p) = \mu_0(p)/n_p = k$, $\mu_0'(q) = \mu_0(q)/n_q = l$.

Hence, $\mathbb{C}(PN)$ is an input-ordinary CPN code.

The remaining cases are that there exists $a \in X_p$ such that $n_q \nmid W(q, a)$ or $n_q \nmid W(a, q)$, and that there exists $b \in X_q$ such that $n_p \nmid W(p, b)$ or $n_p \nmid W(b, p)$. By considering the symmetry, we must check the next the next cases:

(A) $\exists a \in X_p$, $\exists b \in X_q$ $[x = W(p, b) > 0, y = W(q, a) > 0$, and $(n_p \nmid x$ or $n_q \nmid y)]$
(B) $\exists a \in X_p$, $\exists b \in X_q$ $[x = W(b, p) > 0, y = W(q, a) > 0$, and $(n_p \nmid x$ or $n_q \nmid y)]$
(C) $\exists a \in X_p$, $\exists b \in X_q$ $[W(b, p) = 0, y = W(q, a) > 0$, and $n_q \nmid y]$
(D) $\exists a \in X_p$, $\exists b \in X_q$ $[x = W(b, p) > 0, y = W(a, q) > 0$, and $(n_p \nmid x$ or $n_q \nmid y)]$
(E) $\exists a \in X_p$, $\exists b \in X_q$ $[x = W(b, p) > 0, W(q, a) = 0$, and $n_p \nmid x]$

By LEMMA 5.2.1,5.2.2, 5.2.3 and 5.2.5, we can show that $C$ is an input-ordinary CPN code in case of (A), (B), (C) or (E), respectively. On the other hand the case $(D)$ does not happen because $C$ is not a maximal CPN code by LEMMA 5.2.4. □

We state the LEMMA 5.2.1∼5.2.5 in referred in the proof of THEOREM 5.2.1.

**LEMMA 5.2.1** Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$ which is satisfied the following condition (A). If $C = \mathbb{C}(PN) \neq \emptyset$ is a maximal CPN code, then it is a uniform code $X^k$, that is, an input-ordinary CPN code.

$(A)$ $\exists a \in X_p$, $\exists b \in X_q$ $\{x = W(p, b) > 0, y = W(q, a) > 0$, and $(n_p \nmid x$ or $n_q \nmid y)\}$.

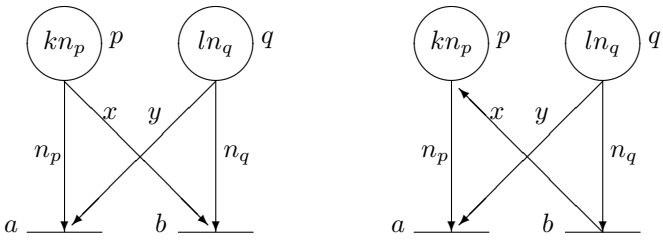
(Proof) By THEOREM 5.1.1, $0 < x \leq n_p$ and $0 < y \leq n_q$ hold, while $x \neq n_p$ or $y \neq n_q$ holds by the hypothesis (A).

Without the loss of generality, we may suppose $0 < x < n_p$ and $0 < y \leq n_q$.

Note that both $\delta(\mu_0, a^{k-1}) = (n_p, ln_q - (k-1)y)$ and $\delta(\mu_0, b^{l-1}) = (kn_p - (l-1)x, n_q)$ must be positive markings because $a \in X_p$ and $b \in X_q$. So the following equalities holds:

$$\delta(\mu_0,\, a^{k-1}b) = (n_p - x,\, (l-1)n_q - (k-1)y)$$
$$\delta(\mu_0,\, b^{l-1}a) = ((k-1)n_p - (l-1)x,\, n_q - y)$$

Since $C$ is a maximal prefix code and $0 < n_p - x < n_p$, $(l-1)n_q - (k-1)y = 0$ must hold by the first equality above. Moreover $y \leq n_q$ implies $l \leq k$. Similarly since $C$ is a maximal prefix code, the assumption $0 < n_q - y < n_q$ implies that $k < l$ because $(k-1)n_p - (l-1)x = 0$ and $n_p > x$. This is a contradiction.




($a$) Condition (A) ($b$) Condition (B)

Figure 5.5: Arc Connection under Condition (A) or (B).

Therefore we have $y = n_q$ and $l = k$. So the code $C = \mathbb{C}(PN)$ is the uniform code $X^k$, that is an input-ordinary CPN code. □

**LEMMA 5.2.2** Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$ which is satisfied the following condition (B). If $C = \mathbb{C}(PN) \neq \emptyset$ is an maximal CPN code, then it is an input-ordinary CPN code.

$(B)\, \exists a \in X_p,\, \exists b \in X_q\, [x = W(b,p) > 0, y = W(q,a) > 0, \text{ and } (n_p \nmid x \text{ or } n_q \nmid y)]$.

(Proof) By THEOREM 5.1.1, $0 < y \leq n_q$ must hold. Moreover by the condition (B), $n_p \nmid x$ or $y \neq n_q$ also holds.

At first assume that $k > 1$ and $l = 1$. Suppose that $0 < y < n_q$. $n_q - y$ tokens remain in $q$ after $a$ fires once. Then since $b$ is not enable, this contradicts to the maximality of the code $C$. Therefore $y = n_q$ must hold. Then $C$ is a uniform code $X$, that is an input-ordinary CPN code.

Next assume that $l > 1$. Since $\delta(\mu_0, b^{l-1}a) = ((k-1)n_p + (l-1)x, n_q - y)$ holds, $0 < y < n_q$ implies that $b$ is not enable at this marking. This is a contradiction to the maximality of $C$. Therefore we have $y = n_q$ and $n_p \nmid x$. $k < l$ must hold because $a \in X_p$.

Since $\delta(\mu_0, ba^j) = ((k-j)n_p + x, (l-1-j)n_q)$ is not positive for some $j$, $j = l-1, (k-j)n_p + x > 0$. Therefore

$$k + \frac{x}{n_p} + 1 > l > k. \tag{5.1}$$

We define the set $Y$ of transitions as follows:

$$Y = \{b_i \in X | n_p \nmid W(b_i, p)\}.$$

Since each element $b_i \in Y$ is $b_i \notin X_p$, we get $b_i \in X_q$, that is, $Y \subset X_q$. As mentioned above at the inequality (5.1), $b_i \in Y$ implies that the inequality $k + (W(b_i,p)/n_p) + 1 > l > k$ must hold. After $b_i$ fires once, since the number

of tokens in $p$ is not a multiple of $n_p$, for any transition $c \in X$, $K_{b_i}(c) = \{q\}$, that is, $W(q,c) = n_q$ and $W(c,q) = 0$ hold (This means that there is no arc coming into $q$).

Setting $t = l-k-1$, note that $0 \leq tn_p < W(b_i,p)$. Let $PN' = (P, X, W', \mu_0)$ be the Petri net obtained by replacing the weight $W(b_i,p)$ of each arc $(b_i,p) \in Y \times \{p\}$ with $tn_p$, that is,

$$W'(b_i,p) = tn_p \quad b_i \in Y \subset X.$$

Each weight of the other arcs is unchanged. And let $\delta$ and $\delta'$ be the transition functions of $PN$ and $PN'$ respectively.

If $w \in C \setminus X^*YX^*$, then $\delta(\mu_0,u) = \delta'(\mu_0,u)$ for any $u$ with $w = uv (u, v \in X^*)$. On the other hand, If $w \in C \cap X^*YX^*$, then for $u$ with $w = uv$,

$$\begin{aligned}
&u \in (X \setminus Y)^*, v \in X^*YX^* \Rightarrow \delta(\mu_0,u)(p) = \delta'(\mu_0,u)(p) > 0\\
&u \in X^*YX^*, v \in X^+ \Rightarrow\\
&\qquad \delta(\mu_0,u)(p) > \delta'(\mu_0,u)(p) > kn_p + (l-k-1)n_p - (l-1)n_p = 0\\
&u = w, v = 1 \Rightarrow\\
&\qquad \delta(\mu_0,w)(p) > \delta'(\mu_0,w)(p) \geq kn_p + (l-k-1)n_p - (l-1)n_p = 0
\end{aligned}$$

and $\delta(\mu_0,u)(q) = \delta'(\mu_0,u)(q)$. Moreover we have $\delta(\mu_0,w)(q) = \delta'(\mu_0,w)(q) = 0$. These yeild $\mathbb{C}(PN) = \mathbb{C}(PN')$. While in the Petri net$(P, X, W',\ \mu_0)$, $W'(p,c)$, $W'(c,p)$ and $W'(c,q)$ are multiples of $n_p$, $W'(q,c) = n_q$ for any $c \in X$. Therefore $C(P, X, W',\ \mu_0)$ is an input-ordinary CPN code, that is, We get the result $\mathbb{C}(PN) = \mathbb{C}(PN') \in$**iCPN**. □

**LEMMA 5.2.3** Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$ which is satisfied the following condition (C). If $C = \mathbb{C}(PN) \neq \emptyset$ is a maximal CPN code,then it is an input-ordinary CPN code.

$$(C)\ \exists a \in X_p,\ \exists b \in X_q \quad [W(b,p) = 0, y = W(q,a) > 0, \text{and}\, n_q \nmid y].$$

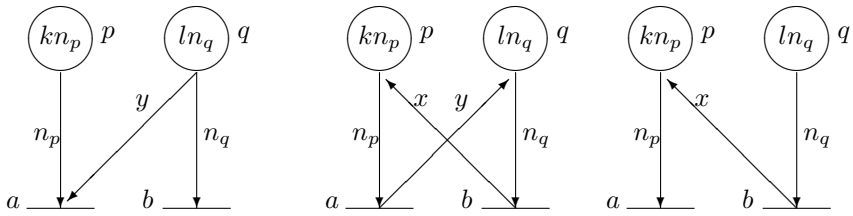
(Proof) We have $0 < y < n_q$ by THEOREM 5.1.1 and the hypothesis (C). Suppose that $k > 1$. $\delta(\mu_0, ab^{l-1}) = ((k-1)n_p, n_q - y)$ and $ab^{l-1}$ is a positive firing sequence, where $\delta$ is the next-state function of $PN$. Then $b$ can not be enable under this marking. Therefore we have $k = 1$ and $l > 1$ because $kl > 1$ is supposed.

Let $c \in X \setminus \{a,b\}$. Noting that $c \in X_p$ or $c \in X_q$ holds, we show that $W(q,c) = n_q$ and $W(c,p) = W(p,c) = 0$ holds if $c \notin X_p$. Then since $c \in X_q$ holds, we have $W(q,c) = n_q$. $W(c,p) > 0$ implies that $cab^{l-2}$ is a positive firing sequence and $\delta(\mu_0, cab^{l-2}) = (W(c,p), n_q - y)$. $b$ can not be enable under this marking. Hence $W(c,p) = 0$ holds. Next we may suppose that $0 \leq W(p,c) \leq n_p$ by THEOREM 5.1.1. $0 < W(p,c) < n_p$ implies that $l = 1$ as discussed above, a contradiction to $kl > 1$. while $W(p,c) = n_p$ implies that $c \in X_p$, a contradiction to $c \notin X_p$. Hence we have $W(p,c) = 0$.

Setting $X_1 = X_p$ and $X_2 = X_q \setminus X_p$,

$$C = (\bigcup_{0 \le i < l} {X_2}^i X_1) \cup {X_2}^l.$$

$C$ is of the same form as EXAMPLE 5.1.2 and is an input-ordinary CPN code. □




(c) Condition (C) (d) Condition (D) (e) Condition (E)

Figure 5.6: Arc Connection under Condition (C), (D) or (E).

**LEMMA 5.2.4** Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$ which is satisfied the following condition (D). If $C = \mathbb{C}(PN)$ cannot be a maximal CPN code.

$(D)\, \exists a \in X_p,\, \exists b \in X_q\, [x = W(b,p) > 0, y = W(a,q) > 0, \text{ and } (n_p \nmid x \text{ or } n_q \nmid y)].$

(Proof) Assume that $k > 1$. There exists some positive integer $i$ such that $ab^i$ is a code word by the maximality of $C$. Then since $ln_q + y - in_q = 0$, $y$ is a multiple of $n_q$. Since $ln_q + y$ tokens remain in $q$ by the firing sequence $a$. $kn_p + x$ tokens remain in $p$ and $(l-1)n_q + y$ tokens remain $q$ by the firing sequence $ab$. Since $aba^j$ is a code word for some positive integer $j$, $x$ is a multiple of $n_p$. This is a contradiction to the hypothesis (D). Symmetrically $l > 1$ concludes a contradiction. □

**LEMMA 5.2.5** Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking $\mu_0$ which is satisfied the following condition (E). If $C = \mathbb{C}(PN) \neq \emptyset$ is a maximal CPN code,then it is an input-ordinary CPN code.

$$(E)\, \exists a \in X_p,\, \exists b \in X_q \quad [x = W(b,p) > 0, W(q,a) = 0, \text{ and } n_p \nmid x].$$

(Proof) Assume that $l > 1$. $kn_p + x$ tokens remain in the place $p$ by the firing sequence $b$. Since the number of tokens in $p$ must become 0 by the firing sequence $ba^i$ for some integer $i$, $x$ must be a multiple of $n_p$. Therefore $l > 1$ is impossible and $l = 1$ holds.

We may assume that $k > 1, l = 1$. For any $c \in X \setminus \{a, b\}$, $\delta(\mu_0, c)(q) = 0$ or $n_q$ holds, where $\delta$ is the next-state function of $PN$. Therefore $W(q, c) = n_q$ or $W(p, c) = n_p$. Setting $X_1 = X_p \setminus X_q$ and $X_2 = X_q$, we have

$$C = (\bigcup_{0 \le i < k} {X_1}^i X_2) \cup {X_1}^k$$

This is an input-ordinary CPN code of the same form as EXAMPLE 5.1.2. □

### 5.2.2 With at least one Source Transitions

In this subsection we show that the code which is generated by a Petri net with two places and at least one source transitions.

**REMARK 5.2.1** A Petri net $PN = (P, X, W, \mu_0)$ is called semi-input-ordinary if the following condition is satisfied.

For each place $p$, there exists a positive integer $n_p$ such that $W(p, a) = 0$ or $= n_p$ and $W(a, p)$ is a multiple of $n_p$ for any transition $a \in X$, and $\mu_0(p)$ is a multiple of $n_p$.

If a Petri net $PN = (P, X, W, \mu_0)$ is semi-input-ordinary, then the code $\mathbb{C}(PN)$ is obviously an input-ordinary CPN code. □

**DEFINITION 5.2.1** Let $PN = (P, X, W, \mu_0)$ be a Petri net. A place $p \in P$ is *controllable* if there are a source transition $c \in X$ and a sink or transform transition $a \in X$ satisfying either of the following two conditions (a) or (b) for any place $q \in P \setminus \{p\}$.

(a) $x > 0$ and $xv - uy > 0$,

(b) $x > 0$, $u > 0$, $y^* > 0$ and $v = 0$,

where $x = W(p, a)$, $y = W(q, a)$, $u = W(c, p)$, $v = W(c, q)$ and $y^* = W(a, q)$. Otherwise $p$ is called *uncontrollable*. □

For example, in case of $|P| = 2$, there are the fifteen ways to give weights on the arcs among arbitrary two places $p$ and $q$, a source transition $c \in X$ and a sink or transfrom transition $a \in X$.

**FACT** Let $u$ and $x$ be nonnegative integers and $d = gcd(u, x)$ be the greatest common divisor of $u$ and $x$ (note that $gcd(0, x) = x$ and $gcd(u, 0) = u$). Then $us + xt = d$ for some integers $s$ and $t$. □

**LEMMA 5.2.6** Let $PN = (P, X, W, \mu_0)$ be a Petri net with source transitions. If a place $p \in P$ be controllable, then the following conditions hold:

For arbitrary nonnegative integers $i$ and $j$ with $\mu_0(p) - d \times i \ge 0$, there is a word $w \in X^+$ such that

$$\begin{aligned} &\delta(\mu_0, w)(p) = \mu_0(p) - d \times i, \\ &\delta(\mu_0, w)(q) \ge \mu_0(q) + j, \qquad \text{for} \quad \forall q \in P \setminus \{p\}. \end{aligned}$$

(Proof) We prove the claim, separating it into the case (1) $i = 1$ and the case (2) $i > 0$ in the sequel.

(1) Let $q$ be an arbitrary place and $q \neq p$. Since $p$ is controllable, there are two transition $c$ and $a$ satifying either the condition (a) or (b) in DEFINITION 5.2.1. Set $x = W(p, a)$, $y = W(q, a)$, $u = W(c, p)$, $v = W(c, q)$ and $y^* = W(a, q)$.

At first assume that (a) $x > 0$ and $xv - uy > 0$.

$$\delta(\mu_0, c^x a^u)(p) = \mu_0(p) + ux - xu = \mu_0(p),$$
$$\delta(\mu_0, c^x a^u)(q) = \mu_0(q) + vx - yu \geq \mu_0(q) + 1,$$

where $\delta$ is the next-state function of $PN$. Next if (b) $x$, $u$, $y^* > 0$ and $v = 0$ hold, then $\delta(\mu_0, c^x a^u)(p) = \mu_0(p)$ and $\delta(\mu_0, c^x a^u)(q) = \mu_0(q) + y^* u \geq \mu_0(q) + 1$. Therefore, choosing $w = (c^x a^u)^j$, the claim holds.

(2) By FACT stated above, $us_0 + xt_0 = d$ holds for some integers $s_0$ and $t_0$ where $d = gcd(u, x)$. For any positive integer $k$ ,

$$u(-s_0 + kx) + (-x)(t_0 + ku) = -d,$$
$$v(-s_0 + kx) + (-y)(t_0 + ku) = v(-s_0) + (-y)t_0 + k(vx - uy).$$

For each place $q \in P$, we can take $k = k_q$ such that $-s_0 + k_q x > 0, t_0 + k_q u > 0$ and $v(-s_0) + (-y)t_0 + k_q(vx - uy) > 0$. and $k_0 = max\{k_q | q \in P, q \neq p\}$. Moreover put $s = -s_0 + k_0 x$, $t = t_0 + k_0 u$ and $w_1 = c^s a^t$. Then $\delta(\mu_0, w_1)(p) = \mu_0(p) - d$ and $\delta(\mu_0, w_1)(q) = \mu_0(q) + v(-s_0) + (-y)t_0 + k_0(vx - uy) \geq \mu_0(q) + 1$.

By the case (1), for any positive integer $j$, there is some word $w_2$ such that

$$\begin{aligned} &\delta(\mu_0, {w_1}^i w_2)(p) = \mu_0(p) - d \times i, \\ &\delta(\mu_0, {w_1}^i w_2)(q) > \mu_0(q) + i + j \geq \mu_0(q) + j, \end{aligned} \tag{5.2}$$

where $i$ satifying $\mu_0(p) - d \times i \geq 0$.

Secondly if (b) $x$, $u$, $y^* > 0$ and $v = 0$, then $\delta(\mu_0, w_1)(p) = \mu_0(p) - d$ and $\delta(\mu_0, w_1)(q) = \mu_0(q) + y^* s > \mu_0(q) + j$. Similarly the relation (5.2) can be concluded. Therefore the proof is completed. □

The next theorem holds regardless of the number $|P|$ of places.

**LEMMA 5.2.7** Let $PN = (P, X, W, \mu_0)$ be a Petri net with source transitions and $\mu_0$ be a positive marking. Let $C = \mathbb{C}(PN)$ be a maximal CPN code. If a place $p \in P$ is controllable, the following conditions hold.

(1) There exists some positive integer $n_p$ such that $W(p, a) = 0$ or $W(p, a) = n_p$ for any $a \in X$.

(2) $n_p | W(a, p)$ for any $a \in X$ .

(3) $n_p | \mu_0(p)$.

(Proof) Since $p \in P$ is a controllable place, there exist a source transition $c \in X$ and a sink or transform transition $a \in X$. Set $u = W(c, p)$, $x = W(p, a)$ and $d = gcd(u, x)$.

By LEMMA 5.2.6, for any positive integer $i$, there is some word $w \in \{a, c\}^+$,

$$\delta(\mu_0, w)(p) = \mu_0(p) - i \cdot d \geq 0,$$

where $\delta$ is the next-state function of $PN$. $0 < \mu_0(p) - id < x$ is impossible because $C = \mathbb{C}(PN)$ is a maximal CPN code. Moreover $d|x$ yields $x = d$ and set $n_p = x = d$. Therefore we can write $\mu_0(p) = kn_p$ for some positive integer $k$. The condition (3) holds.

We can choose $w' \in \{a, c\}^+$ and an enough large integer $N$ such that

$$\begin{aligned} &\delta(\mu_0, w')(p) = n_p, \\ &\delta(\mu_0, w')(q) > N \quad \forall q \in P \setminus \{p\}. \end{aligned}$$

Let $p\bullet = \{a_1, a_2, \cdots, a_n\}$ with $a_1 = a$ and $x_i = W(p, a_i) > 0$ for $i\,(1 \leq i \leq n)$ $x_i \leq n_p$ must hold because $\delta(\mu_0, w'a_i)$ is defined. Moreover $x_i = n_p$ holds, because $0 < x_i < n_p$ implies that $\delta(\mu_0, w'a_i)(p) = n_p - x_i$ and $\delta(\mu_0, w'a_ia)$ is undefined. We can obtain $W(p, a_i) = n_p$ for any $a_i \in p\bullet$, namely the condition (1) holds.

And $n_p \nmid u$ implies that $\delta(\mu_0, w'ca^i)$ is undefined for some integer $i$. This yeilds a contradictoin. This means the condition (2) holds. □

**COROLLARY 5.2.1** Let $PN = (P, X, W, \mu_0)$ be a Petri net with at least one source transitions and $\mu_0$ be a positive marking. If each $p \in P$ is controllable and $C = \mathbb{C}(PN)$ is a maximal CPN code, then $C$ is an input-ordinary CPN code.

Note that the COROLLARY5.2.1 is true independently to the number of places. Then we check the case that both two places are uncontrollable and the case that one place is controllable but the other is not. The remaining cases needs the restriction that the number of places is two.

**LEMMA 5.2.8** Let $PN = (P, X, W, \mu_0)$ be a Petri net with at least one source transitions, $\mu_0$ be a positive marking and $|P| = 2$ $(P = \{p, q\})$. If each place is uncontrollable and $C = \mathbb{C}(PN)$ is a maximal CPN code, then $C$ is an input-ordinary CPN code.

(Proof) Assume that there exists a source transition $c$ such that $u = W(c, p) = 0$ and $v = W(c, q) > 0$ Since $\delta(\mu_0, w)(p) = 0$ for some $w \in C$, there exists a sink or transform transiton $a$ with $x = W(p, a) > 0$. Then,

$$vx - uW(q, a) = vx > 0.$$

This means that $p$ is controllable, a contradiction. Therefore each source transition $c$ has just two output arcs, that is, $c\bullet = \{p, q\}$.

So let $c$ be a source transition with $u = W(c, p) > 0$ and $v = W(c, q) > 0$. We first show that for any sink or transform transition $a \in X$,

$$vx - uy = 0, \quad x > 0, \quad y > 0,$$

where $x = W(p, a)$ and $y = W(q, a)$. Note that $a$ must be a sink transition. We may suppose x¿0 because $a$ is a sink or transform transition. Since $p$ is

uncontrollable, $vx - uy \leq 0$ and $y > 0$ hold. Again by the uncontrallability of $q$, we have $vx - uy \geq 0$, and $vx - uy = 0$ in the sequel.

Let $a$ and $b$ be any sink transitions, and set $x = W(p,a)$, $y = W(q,a)$, $x' = W(p,b)$, $y' = W(q,b)$. Since $(v,u)$ is a nontrivial solution of

$$\begin{pmatrix} x & y \\ x' & y' \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

we have $xy' - x'y = 0$, that is,

$$\frac{x'}{x} = \frac{y'}{y} = r$$

for some positive rational number $r$.

$p \in K(a)$ or $q \in K(b)$ holds, where $K$ is the critical place mapping of $PN$. We may assume that $p \in K(a)$ without loss of generality. Then $\mu_0(p) = kx$ and $\mu_0(q) \geq kx$ for some positive integer $k$. $p \in K(a)$ implies $r \leq 1$ by THEOREM 5.1.1. Consider the marking

$$\mu_1 = \delta(\mu_0, a^{k-1}b) = (x - x', \mu_0(q) - (k-1)y - ry).$$

$r < 1$ implies that $x - x' = (1-r)x > 0$ and $\mu_0(q) - (k-1)y - ry \geq (1-r)y > 0$, that is, $a$ cannot be enabled under the marking $\mu_1$. This is a contradiction. Therefore we have $r = 1, x' = x, y' = y$. $x$ and $y$ are constants determinded by places $p$ and $q$, respectively.

$u = hx$ and $v = hy$ for some positive rational number $h$ because $vx - uy = 0$. Consider the marking

$$\mu_2 = \delta(\mu_0, ca^{k+\lfloor h \rfloor}) = ((h - \lfloor h \rfloor)x, \mu_0(q) - ky + (h - \lfloor h \rfloor)y),$$

where $\lfloor h \rfloor$ is meant the greatest integer not over $h$. If $h \neq \lfloor h \rfloor$, then $a$ cannot be enabled under the marking $\mu_2$. Therefore $h = \lfloor h \rfloor$ is an integer. For each souce transition $c$, there exists a positive integer $h_c$ such that $W(c,p) = h_c x$ and $W(c,q) = h_c y$.

Hence $C = \mathbb{C}(PN) = \mathbb{C}(PN')$, where $PN'$ is the following input-ordinary Petri net $(P', X', W', \mu_0')$:

$$\begin{array}{ll} P' = P = \{p,q\},\ X' = X, & \\ \mu_0(p) = k, \mu_0(q) \geq k, & \\ W'(c,p) = W'(c,q) = h_c & \text{if } c \text{ is a source transition} \\ W'(p,a) = W(q,a) = 1 & \text{if } a \text{ is a sink transition.} \end{array}$$

□

**LEMMA 5.2.9** Let $PN = (P, X, W, \mu_0)$ be a Petri net with source transitions, $\mu_0$ be a positive marking and $|P| = 2$ ($P = \{p,q\}$). One place $p$ is controllable and the other place $q$ is not. If $C = \mathbb{C}(PN)$ be a maximal CPN code, then $C$ is an input-ordinary CPN code.

(Proof) If there are no transition $b \in X$ with $K(b) = \{q\}$, where $K$ is the critical place mapping of $PN$, then we can get the same result as the case of $|P| = 1$. So we may suppose that there exists at least one transition $b$ with $K(b) = \{q\}$. Since $p$ is controllable, by LEMMA 5.2.7, there exists a positive integer $n_p$ such that $W(p,a) \in \{0, n_p\}$ and $n_p | W(a,p)$ for any transition $a \in X$ and $\mu_0(p) = kn_p$ for some $k$.

Let $b$ be a transition with $K(b) = \{q\}$ and $a$ be an arbitrary transition. Set $n = \mu_0(q)$, $n_q = W(q,b) > 0$, $z = W(q,a) \geq 0$.

At first we show that the weight $z$ equals either $n_q$ or 0. We can write $n = ly$ for some positive integer $l$ because $K(b) = \{q\}$. Since $\delta(\mu_0, b^{l-1}a) = (k'n_p, n_q - z)$ and $k' \geq 1$, $z = 0$ or $z = n_q$.

By the way, we define the set $Y$ of transitions by $Y = \{d_i \in X | u_i = W(d_i, q), n_q \nmid u_i\}$. If $Y = \emptyset$, the proof is completed because $PN$ becomes a semi-input-ordinary Petri net. So we may suppose that $Y \neq \emptyset$. Next we show that (1) for $\forall a \in X, \forall d_i \in Y, \forall w \in X^* \setminus X^*YX^*$ $\delta(\mu_0, wd_i)$ is defined $\Rightarrow K_{wd_i}(a) = \{p\}$ and (2) for $\forall a, t \in X, \forall w \in X^*$ $\delta(\mu_0, wt)$ is defined $K_w(a) = \{p\} \Rightarrow K_{wt}(a) = \{p\}$ We prove the implication (1). It is easily check that $\delta(\mu_0, w) = (k'n_p, l'n_q)$ for some positive integers $k'$ and $l'$ because $w$ does not contain any element in $Y$. Since $\delta(\mu_0, wd_i) = (k'n_p - \Delta_i n_p, l'n_q + u_i)$ where $\Delta_i n_p = W(p, d_i) - W(d_i, p)$ and $l'n_q + u_i$ isn't multiple of $y$, we get $K_{wd_i}(a) = \{p\}$ and moreover $W(p,a) = n_p$, The following inequality must be satisfied:

$$\frac{k'n_p - \Delta_i n_p}{n_p} < \frac{l'n_q + u_i}{n_q}$$
$$k' < l' + \frac{u_i}{n_q} + \Delta_i$$

Now we prove the implication (2). Setting $\delta(\mu_0, w) = (kn_p, n)$ with $kn_q < n$, because $p$ is controllable and $K_w(a) = \{p\}$.

First of all, let $t$ be a sink transition. $W(p,t) = n_p > 0$ because $p$ is controllable and $W(d_i, q) = u_i$ is not multiple of $y$ for some $d_i \in Y$. Moreover setting $W(q,t) = x$, $0 \leq x \leq n_q$. $\delta(\mu_0, wt) = ((k-1)n_p, n - x)$ and $(k-1)n_q < n - n_q < n - x$. Hence $K_w(a) = K_{wt}(a)$.

Next let $t$ be a transform transition. $W(p,t) = n_p > 0$ and $W(t,q) = x > 0$ because $p$ is controllable. $\delta(\mu_0, wt) = ((k-1)n_p, n + x)$ and $(k-1)n_q < kn_q < n < n + x$. Hence $K_w(a) = K_{wt}(a)$.

Finally let $t$ be a source transition. Since $p$ is controllable, $vn_p - un_q > 0$, where $u = W(t,p)$ and $v = W(t,q)$. $\delta(\mu_0, wt) = (kn_p + u, n + v)$ and

$$\frac{kn_p + u}{n_p} n_q < \frac{kn_p n_q + n_p v}{n_p} = kn_q + v < n + v$$

because $un_q < vn_p$. Hence $K_w(a) = K_{wt}(a)$. The implication (2) has just proved.

This means that once a transition $d_i \in Y$ fires, the place $p$ continues to be a critical place of each transition, namely $K_{wd_iw'}(a) = \{p\}$ for any transition $a \in X$ and any $w' \in X^*$ whenever $\delta(\mu_0, wd_iw')$ is defined.

Here we choose a positive integer $N$ such that $Nn_q > W(d_i, q) = u_i$ for any $d_i \in Y$. Hence It is clear that the equality $C = \mathbb{C}(PN) = \mathbb{C}(PN')$ holds, where $PN'$ is the following input-ordinary Petri net $(P', X', W', \mu_0')$:

$$
\begin{array}{ll}
P' = P = \{p, q\},\ X' = X, & \\
\mu_0(p) = k, \mu_0(q) = l, & \\
W'(p,a) = W(p,a)/n_p \in \{0,1\}, W'(a,p) = W(a,p)/n_p \in \boldsymbol{N}_0 & a \in X, \\
W'(q,a) = W(q,a)/n_q \in \{0,1\}, W'(a,q) = W(a,q)/n_q \in \boldsymbol{N}_0 & a \in X \setminus Y, \\
W'(q,a) = W(q,a)/n_q \in \{0,1\}, W'(a,q) = N & a \in Y,
\end{array}
$$

This yeilds $C$ is an input-ordinary CPN code. □

The LEMMAs 5.2.7 to 5.2.9 that are stated above derives the next THEOREM 5.2.2

**THEOREM 5.2.2** Let $PN = (P, X, W, \mu_0)$ be a Petri net with source transitions, $\mu_0$ be positive and $|P| = 2$. If $C = \mathbb{C}(PN)$ is a maximal CPN code, then $C$ is an input-ordinary CPN code. □

We obtain the final result of this chapter from the THEOREMs 5.2.1 and 5.2.2.

**THEOREM 5.2.3** Let $PN = (P, X, W, \mu_0)$ be a Petri net, $\mu_0$ be positive and $|P| = 2$. If $C = \mathbb{C}(PN)$ is a maximal CPN code, then $C$ is an input-ordinary CPN code. □

## 5.3 Remarks and Further Works

In this chapter we have proved that **mCPNC** $\subseteq$ **iCPNC** if the number $|P|$ of places in a generating Petri net is $\leq 2$. The problem is still open in case that $|P| > 2$. Our tactics was to apply the above result of the case of $|P| = 2$ to a general Petri net but the application have not been worked well yet though it may be possible.

This problem is important in the following meaning. $\mathbb{C}(N, \mu_0)$ is the set of all nonpositive firing sequences in $(N, \mu_0)$ whose proper prefixes are all positive firing sequences instead. That is, $\mathbb{C}(N, \mu_0) = L \setminus LX^+$, where $L = L(N, \mu_0) \setminus L_+(N, \mu_0)$. $\mathbb{C}(N, \mu_0)$ is a maximal prefix code or $\mathbb{C}(N, \mu_0) = \emptyset$ if and only if all of transitions are enabled under a marking reachable from $\mu_0$ through a positive firing sequence in $(N, \mu_0)$. This condition is obviously true if $(N, \mu_0)$ is input-ordinary. Conversely we wonder whether the set of all positive firing sequences in a general Petri net $(N, \mu_0)$ with $\mathbb{C}(N, \mu_0)$ being a maximal prefix code is identical with the set of all positive firing sequences in some input-ordinary Petri net $(N_1, \mu_1)$, that is, $L_+(N, \mu_0) = L_+(N_1, \mu_1)$.

Extending PROPOSITION 4.4.4 to a Petri net of rank 2, the following typical case holds. Let Petri nets $PN_i\,(1 \leq i \leq k)$ with their rank $\leq 2$ and each $C_i = \mathbb{C}(D_i)$ being a maximal prefix code. Then the composed Petri net code $C = C_1 \otimes C_2 \otimes \ldots \otimes C_k$ must be an input-ordinary CPN code.

The partial problem may be important and effective to restrict conditions, for example,

1. The number of transitions is $\leq k$ for some integer $k$.
2. $\mathbb{C}(PN)$ is finite.
3. The reachable marking of a Petri net is finite.
4. The weighted graph $(V, E)$ where $V$ is the set of places and transitions and $E$ is the set of arcs has no curcuit.

The second and third conditions above may be too strong. These conditions implies the necessary condition that $PN$ has neither a source places nor an isolate places. Even if a Petri net $PN$ has neither a source places nor an isolate places, $\mathbb{C}(PN)$ can be a infinite maximal CPN code. We can easily have a Petri net $PN$ with $\{a^i b^{i+1} \mid i \in \boldsymbol{N}_0\} \subseteq \mathbb{C}(PN)$, which is an infinite and maximal CPN code.

Roughly speaking about the forth condition, if the weight of every outgoing arc from a place $p$ is constant $n_p$ (depending on the place $p$), then the problem seems to become rather easier. It is an important condition that every chain of places $p_i \in P\,(1 \leq i \leq n)$ and transitions $a_i \in X\,(1 \leq i \leq n-1)$

$$p_1, a_1, p_2, a_2, \ldots, p_{n-1}, a_{n-1}, p_n$$

such that $0 < W(p_i, a_i) < n_{p_i}$ and $W(a_i, p_{i+1}) = n_{p_{i+1}}$ has no loop, that is, $p_1, p_2, \ldots, p_{n-1}, p_n$ are all distinct.

In our recent investigation[55], we introduce some dependency of places in a Petri net. This involves the related problem to graph, combinatrics, algebra, and so on.

# Chapter 6

# Conclusion

Recently Petri nets are used not only as technical modeling tools for parallel/concurrent systems, but also as theoretical model of computation like automata, language generators, grammar controllers, and so on. Petri net theory is one of advanced and interested fields in automata, formal languages and computation. In this literature we treated two topics, the Petri net structures (in Chaper 3) and Petri net codes (in Chapters 4 and 5).

In Chapter 3, the notion of automorphism group of a Petri net structure was newly introduced. We showed the main theorem that for a given finite group $G$ there exists a Petri net structure $N$, called a transformation net, such that $\mathbf{Aut}(N)$ is isomorphic to $G$. The structure $N$ corresponds to the right regular representation of $G$.

The four (S-, D-, C-, B-) types of Petri net codes, which are all prefix codes, were introduced as similar way to define Petri net languages. Mainly we use Petri nets as accepters of codes and treat firing sequences themselves without labeling functions. In Chapters 4 and 5, C-type Petri net (CPN, for short) codes are mainly focused. The CPN code $\mathbb{C}(N, \mu_0)$ generated by a Petri net $(N, \mu_0)$ is the set of all nonpositive firing sequences in $(N, \mu_0)$ whose proper prefixes are all positive firing sequences instead. That is, $\mathbb{C}(N, \mu_0) = L \setminus LX^+$, where $L = L(N, \mu_0) \setminus L_+(N, \mu_0)$. If a CPN code is a maximal prefix code, then we call it a maximal CPN code.

In the first half of Chapter 4, various properties of finite maximal CPN codes were investigated and two operations $\oplus$(some kind of parallel operation) and $\otimes$(some kind of interruption) were introduced.

The property being a maximal CPN code over $X$ is not preserved under concatenation, $\oplus$ and $\lambda$-free homomorphism but is preserved under $\otimes$. In the second half, we investigated the generative power of CPN codes. There it is shown that there exists a CPN code which is not context-free, but arbitrary CPN code is a context-sensitive language.

In Chapter 5 we considered the open problem raised in Chapter 4. That is, whether the family **mCPNC** stated above is included in the family **iCPNC** of CPN codes which are generated by some input-ordinary Petri nets.

The notion of maximality of a CPN code is very important in relation to liveness or deadlock. $\mathbb{C}(N, \mu_0)$ is a maximal prefix code or $\mathbb{C}(N, \mu_0) = \emptyset$ if and only if all of transitions are enabled under a marking reachable from $\mu_0$ through a positive firing sequence in $(N, \mu_0)$. This condition is obviously true if $(N, \mu_0)$ is input-ordinary. Conversely we wonder whether the set of all positive firing sequences in a general Petri net $(N, \mu_0)$ with $\mathbb{C}(N, \mu_0)$ being a maximal prefix code is identical with the set of all positive firing sequences in some input-ordinary Petri net $(N_1, \mu_1)$, that is, $L_+(N, \mu_0) = L_+(N_1, \mu_1)$.

We proved that **mCPNC** = **iCPNC** is true in restricted cases, i.e., in the case that the number of places is $\leq 2$, and in the case that the number of transitions is equal to 1. It still remains open in a general Petri net.

# Appendix A

# List of Notations

| | |
|---|---|
| $\boldsymbol{N}$ | set of natural numbers, $\boldsymbol{N} = \{1, 2, 3, \cdots\}$. |
| $\boldsymbol{N}_0$ | set of nonnegative integers numbers, $\boldsymbol{N}_0 = \{0, 1, 2, 3, \cdots\}$. |
| $\boldsymbol{Z}$ | set of integers. |
| $\boldsymbol{R}$ | set of real numbers. |
| $a \leq b$ | $a$ is less(or smaller) than or equal to $b$. |
| $a \geq b$ | $a$ is greater(or bigger) than or equal to $b$. |
| $x \in A$ | $x$ is an element of [belongs to] a set $A$. |
| $a \mid b$ | $a$ is a divisor of $b$. |
| $a \not\mid b$ | $a$ is not a divisor of $b$. |
| $\emptyset$ | empty set. |
| $A \subseteq B$ | $A$ is a subset of $B$. |
| $A \subset B$ | $A$ is a proper subset of $B$. |
| $A \setminus B$ | difference set of $A$ and $B$. |
| $2^P$ | power set of a set $P$. |
| $f : A \to B$ | mapping $f$ from $A$ into $B$. |
| $f : a \mapsto b$ | mapping $f$ maps $a$ into $b$. |
| $B^A$ | set of all mappings from $A$ to $B$. |
| $A \cup B$ | union (or join, cup) of sets $A$ and $B$. |
| $A \cap B$ | intersection (or meet, cap) of sets $A$ and $B$. |
| $A \times B$ | direct (or Cartesian) product of sets $A$ and $B$. |
| $f \circ g$ | composition of mappings $f$ and $g$. |
| $S \simeq T$ | $S$ and $T$ are isomorphic. |
| $\mathfrak{S}_X$ | symmetric group of $X$. |
| $\mathfrak{T}_X$ | full transformation semigroup of $X$. |
| $1$ | unity (of a group), empty sequence, empty word. |
| ${}^tA$ | transpose of a matrix $A$. |

| | |
|---|---|
| $H\backslash G$ | set of all the distinct right cosets of $H$ in $G$. |
| $A^{-1}B$ | $\{x \in M \mid Ax \cap B \neq \emptyset\}$. |
| $AB^{-1}$ | $\{x \in M \mid A \cap xB \neq \emptyset\}$. |
| $X^*$ | free monoid generated by $X$. |
| $X^+$ | free semigroup generated by $X$. |
| $\|x\|$ | length of a word $x$. |
| $u \to v$ | production of $v$ from $u$. |
| $x \Rightarrow_G y$,$x \Rightarrow y$ | 1-step derivation from $x$ to $y$ by a grammar $G$. |
| $x \Rightarrow^* y$ | derivation from $x$ to $y$. |
| $\mathcal{L}(G)$ | language generated by a grammar $G$. |
| $\mathcal{L}(\mathcal{A})$ | language generated by an automaton $\mathcal{A}$. |
| $u \leq_p w$ | $u$ is a prefix of $v$. |
| $u \leq_s w$ | $u$ is a suffix of $v$. |
| $u <_p w$ | $u$ is a proper prefix of $v$. |
| $u <_s w$ | $u$ is a proper suffix of $v$. |
| $\diamond$ | shuffle product of two words/languages. |
| $L^\diamond$ | shuffle closure of a language $L$. |

| | |
|---|---|
| $\delta_{PN}(\mu, t)$ or $\delta(\mu, t)$ | next-state function of a Petri net $PN$. |
| $\mu_0[\sigma>\mu_n$ | $\mu_n$ is reachable from $\mu_0$ by $\sigma$. |
| $R(N, \mu_0)$ or $R(\mu_0)$ | set of all possible markings reachable from $\mu_0$ in a Petri net $(N, \mu_0)$. |
| $L(N, \mu_0)$ or $L(\mu_0)$ | set of all possible firing sequences from $\mu_0$ in a Petri net $(N, \mu_0)$. |
| $L_+(N, \mu_0)$ or $L_+(\mu_0)$ | set of all possible positive firing sequences from $\mu_0$ in a Petri net $(N, \mu_0)$. |
| $d(t_1, t_2)$ | synchronic distance between two transitions $t_1$ and $t_2$. |
| $\#_\sigma(t)$ | number of occurrences of $t$ in $\sigma$. |
| $Tr(\sigma)$ | $\{t \mid t \in T, \#_\sigma(t) > 0\}$, set of transitions occurring in $\sigma$. |
| $\Delta(t)$ | displacement of transition $t$. |
| $\Delta(\sigma)$ | $\Sigma_{i=1}^{k}\Delta(s_i)$ for a firing sequence $\sigma = s_1 s_2 \dots s_k$ $(s_i \in T)$. |
| $\bullet t$ | input places of a transition $t$. |
| $t \bullet$ | output places of a transition $t$. |
| $\bullet p$ | input transitions of a place $p$. |
| $p \bullet$ | output transitions of a place $p$. |
| $\mathrm{Stab}(\mu_0)$ | $\mathrm{Stab}(\mu_0) = \{w \mid w \in L(\mu_0) \text{ and } \delta(\mu_0, w) = \mu_0\}$. |
| $\mathbb{S}(N, \mu_0)$ | base of $\mathrm{Stab}(\mu_0)$. |
| $\mathbb{D}(N, \mu_0)$ | set of all positive firing sequences of $\mathbb{S}(N, \mu_0)$. |
| $\mathbb{C}(N, \mu_0)$ | $\{w \mid w \in L(\mu_0) \setminus L_+(\mu_0), u \in L_+(\mu_0), w = uv, u \neq w\}$. |
| $\mathbb{B}(N, \mu_0)$ | $\mathbb{C}(N, \mu_0) \setminus \mathbb{D}(N, \mu_0)^* X^*$. |
| $\mathbf{Aut}(N)$ | group of all automorphisms of a net $N$. |
| $\mathbf{1}_P$ | identity mapping on $P$. |
| $K_w$ | critical place mapping of a Petri net. |

□ Q.E.D, q.e.d., quod erat demonstrandum.

# Appendix B

# Ramark on Contributors

This dissertation is developed on the basis of joint research with Professor Toshimutsu Inomata, Professor Masami Ito and Professor Genjiro Tanaka. And this is certified in the enclosed declarations.

| Papers | Chapter | Prof. Inomata | Prof. Ito | Prof. Tanaka |
|---|---|---|---|---|
| [52] | 3 | 30% | 0% | 40% |
| [51] | 3 | 30% | 0% | 40% |
| [37] | 4 | 0% | 50% | 0% |
| [38] | 4 | 0% | 50% | 0% |
| [54] | 5 | 0% | 0% | 0% |
| [55] | 5 | 0% | 0% | 0% |

This declaration certifies that the contribution of Yoshiyuki Kunimochi to our joint papers in the following:

| Papers | Chapter | Yoshiyuki Kunimochi |
|---|---|---|
| [52] | 3 | 30% |
| [51] | 3 | 30% |
| [37] | 4 | 50% |
| [38] | 4 | 50% |
| [54] | 5 | 100% |
| [55] | 5 | 100% |

# Index

# Bibliography

[1] B.Acu and W. Reisig. Compensation in workflow nets. *ICATPN2006, LMCS*, 4024:65–83, 2006.

[2] J. Berstel and D. Perrin. *Theory of Codes*. Pure and Applied Mathematics. Academic Press, 1985.

[3] J.A. Brzozowski. Roots of star events. *J. ACM*, 14(3):466–477, 1967.

[4] C.A.Petri. Interpretations of net theory. Technical report, St. Augustin, Cesellshaft fur Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-75-07, Second Edition, 1976.

[5] A.H. Clifford and G.B. Preston. *The Algebraic Theory of Semigroups*, volume 1 of *American Mathematical Society*. Providence R.I., 1961.

[6] F. Commoner. Deadlocks in petri nets. Technical report, Wakefield, Applied Data Research, Inc. Report #CA-7206-2311, 1972.

[7] C.Rackoff. The covering and boundedness problems for vector addition systems. *Theoret. Comput. Sci.*, 6:223–231, 1978.

[8] A. de Luca and S. Varricchio. *Finiteness and Regularity in Semigroups and Formal Languages*. Monographs on Theoretical Computer Science · An EATCS Series. Springer, July 1999.

[9] J. B. Dennis and S. S. Patil. Speed independent asynchronous control structures. *in Proc. 4th Hawaii Int. Conf. Syst. Sci.*, pages 55–58, 1971.

[10] R. Devillers E. Best and M. Koutny. *Petri Net Algebra*. Monographs in Theoretical Computer Science · An EATCS Series. Springer, 2001.

[11] F. Gécseg and I Peák. *Algebraic Theory of Automata*. Akadémiai Kiadó, Budapest, 1972.

[12] S. Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, New York, 1966.

[13] S. Ginsburg. *Algebraic and Automata-Theoritic Properties of Formal Languages*. North-Holland, Amsterdam, 1975.

[14] M. Hack. The recursive equivalence of the reachability problem and the liveness problem for petri nets and vector addition systems. *FOCS*, pages 156–164, 1974.

[15] M. Hack. *Decidability Question for Petri Nets.* Ph.D. dissertation, Dept. of Electrical Engineering, MIT, 1975.

[16] M. Hack. Petri net languages. Project, mac, Comp. Struct. Group Memo 124, Project MAC, MIT, 1975.

[17] M. Hack. The equality problem for vector addition system is undeciable. *Theoret. Comput. Sci.*, 2:77–95, 1976.

[18] H.J.Shyr. *Free monoids and Languages, Lecture Notes.* Hon Min book Company, Taichung, Taiwan, 1991.

[19] J. Hopcroft and J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoret. Comput. Sci.*, 8(2):135–159, 1979.

[20] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading MA, 1979.

[21] M. Ito. *Algebraic Theory of Automata and Languages.* World Scientific, 2004.

[22] J.L.Peterson. Computation sequence sets. *Journal of Computer and System Sciences*, 13(1):1–24, August 1976.

[23] J.L.Peterson. *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, New Jersey, Prentice Hall, Inc., 1984.

[24] J.M.Howie. *Fundamentals of Semigroup Theory.* London Mathematical Society Monographs New Series 12. Oxford University Press, 1995.

[25] R. Karp and R. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3:167–195, 1969.

[26] R. Keller. Vector replacement systems: A formalism for modelling asynchronous systems. Technical Report 117, Computer Science Lab., Princeton University, 1972.

[27] S. R. Kosaraju. Decidability of reachability in vector addition systems. In *14th Annual ACM Symp. Theory Computing*, pages 267–281, San Francisco, May 1982.

[28] Y. Kunimochi and T. Inomata. The reachability of a petri net representing a residue class group. *TECHNICAL REPORT OF IEICE*, COMP95-71:47–54, Dec. 1995.

[29] K. Lautenbach. Liveness in petri nets. Technical report, St. Augustin, Cesellshaft fur Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-75-02.1, 1975.

[30] R. Howell L.Cherkasove and L. Rosier. Bounded self-stabilizing petri nets. *Acta Infomatica*, 32:189–207, 1995.

[31] L.H.Landweber and E.L.Robertson. Properties of conflict free and persistent petri nets. *J. ACM*, 25(3):352–364, 1978.

[32] K. Lodaya. Petri nets event structures and algebla. *Formal Models, Languages and Applications, Machine Perception Artificial Intelligence*, 66:246–259, 2006.

[33] M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1983.

[34] M. Lothaire. *Algebraic Combinatorics on Words*. Encyclopedia of Mathematics and its Applications 90. Cambrige University Press, 1990.

[35] E. W. Mayr. An algorithm for the general petri net reachability problem. *SIAM, J. Comput.*, 13(3):441–460, Aug. 1984.

[36] D. P. Misunas. Petri nets and speed independent design. *Comm. ACM*, 16(8):474–481, 1973.

[37] M.Ito and Y.Kunimochi. CPN languages and codes. *Technical Report kokyuroku, RIMS, Kyoto University*, 1222:46–49, 7 2001.

[38] M.Ito and Y.Kunimochi. Some petri nets languages and codes. *Lecture Notes in Computer Science*, 2295:69–80, 2002.

[39] Tadao Murata. Petri nets: Properties, analysis and application. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[40] J.L. Peterson. *Modeling of Parallel Systems*. Ph.D. dissertation, Department of Electrical Engineering, Stanford University, Stanford, California, 1973.

[41] C. A. Petri. *Kommunikation mit Automaten*. Ph.D. dissertation, Rheinisch-Westfalisches Institut fur. Instrumentelle Mathematik an der Universitat Bonn, Bonn, 1962.

[42] C.V. Ramamoorthy and G.S.Ho. Performance evaluation of asynchronous concurrent systems using petri nets. *IEEE Trans. Software Eng.*, SE-6(5):440–449, Sept. 1980.

[43] G. Rozenberg and A. Salomaa. *Handbook of Formal Languages, Vol.1 WORD, LANGUAGE, GRAMMAR*. Springer, 1997.

[44] A. Salomaa. *Computation and Automata*. Cambridge University Press, 1985.

[45] M.P. Schützenberger. On an application of semigroup method to some problems in coding. *IRE, Trans. Information Theory*, IT-2:47–60, 1956.

[46] J. Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press, 2008.

[47] G. Tanaka. Prefix codes determined by petri nets. *Algebra Colloquim*, 5:255–264, 1998.

[48] T.Chatain and C. Jard. Complete finite prefixes of symbolic unfoldings of safe time petri nets. *ICATPN2006, LMCS*, 4024:124–145, 2006.

[49] T.Murata and Z.Wu. Fair relation and modified synchronic distances in a petri nets. *J. Franklin Inst.*, 320(2):63–82, Aug. 1985.

[50] U.Coltz and Y.Chong-Yi. Synchronic structure—a tutirial. *Lecture Notes in Computer Science*, 222:233–252, 1986.

[51] T. Inomata Y. Kunimochi and G. Tanaka. Automorphism groups of transformation nets (in Japanese). *IEICE Trans. Fundamentals*, J79-A,(9):1633–1637, Sep. 1996.

[52] T. Inomata Y. Kunimochi and G. Tanaka. On automorphism groups of nets. *Publ. Math. Debrecen*, 54 Supplement:905–913, 1999.

[53] Hsu-Chun Yen. Introduction to petri net theory. *Studies in Computational Intelligence(SCI)*, 25:343–373, 2006.

[54] Y.Kunimochi. Some structures of maximal prefix codes generated by petri nets. *Technical Report kokyuroku, RIMS, Kyoto University*, 1503:139–147, 2006.

[55] Y.Kunimochi. Place dependency of a petri net generating a maximal prefix code. *Technical Report kokyuroku, RIMS, Kyoto University*, 1604:80–89, 2008.

[56] Shyr-Shen Yu. *Language and Codes.* Tsang Hai Book Publishing Co., 2005.

**Debrecen University** April 22, 2009

# APPLICATION FORM

To obtain doctorial (Ph.D.) degree

Individual

I Personal Information

Name: Yoshiyuki Kunimochi

Birth Place: Shizuoka, Japan

Birth Day: 1964/8/18

Mother's Name: Nagata Masako

ID Number: TH7991773 (Passport No)

Address : Bright-town Kakegawa 504, 1-18-4 Chuou, akegawa-Shi, 436-0056, Japan

Postal Address: Bright-town Kakegawa 504, 1-18-4 Chuou, Kakegawa-Shi, 436-0056,Japan

Office Address: Shizuoka Institute of Science and Technology, 2200-2 Toyosawa, Fukuroi-shi, 437-855, Japan

Degree Subject of Diploma: Engineer of Informatics Quality: Non Available

Institution of Grant Certification: Shizuoka University no/year: 2232/1989

II Date of scientific work

Published: 20 Professional Paper: 20 recension 5 others -

III Knowledge of language and its level English(excellent), Shizuoka University language exam, 1985

IV Scientific subject of doctoral degree Informatics

Discipline: Theoretical Informatics Doctoral School Informatics (Debrecen University, Faculty of Informatics)

Doctoral Program: Foundation of Informatics

Title of dissertation: Algebraic Properties of Petri net Languages and Codes

Supervisor: Dr. Pál Béla Dömösi

Requested subject of examination of doctorate: Foundation of Informatics(main subject), Automata and Formal Languages(second subject)

Attachments: number of

.....

Sign of Applicant

Debrecen, April 22, 2009