Algebraic Properties of Petri Net Languages and Codes

Ph.D. Thesis

Yoshiyuki Kunimochi

Supervisor: Prof. Pál Dömösi

University of Debrecen
Doctoral Committee of Natural Sciences
Doctoral School of Computer Sciences

Debrecen, 2009

Contents

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.

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 language over an alphabet X is defined as a subset of the free monoid X^* generated by X. A Language of our interest is mainly determined by some procedure that is, computation and derivation, and so on. By applying the concept of a automaton to a Petri net, the Petri net generates a language, 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.

Recently many classes of languages based on Petri nets have been eagerly devised and investigated. For example, some regulated grammars with Petri nets are introduced. Their powers are interesting in the sense that they are often distinct from the classical language classes.

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 of prefix codes based on Petri nets. He named them an S-type, a D-type, a C-type and a B-type Petri net code, respectively[47].

Chapter 2 plays roles 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 notations. Next, we explain the basic concepts of automata and formal languages. After then, 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. The problem of automorphism groups of nets is described. 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 C-type Petri net codes (CPN code, for short) introduced in Chapter 2. This is a joint work[38] with Professor Masami Ito. A CPN code is the set of all minimal sequences with respect to the prefix order among the firing sequences through which the state reach from the positive initial marking to a nonpositive marking . A CPN code of course becomes a prefix code. A CPN code is called a maximal CPN code if it is a maximal prefix code. In a Petri net which generates a maximal CPN code, each transition is enable iff every reachable marking is positive. 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.

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 obviously a maximal CPN code. The problem 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 ≤ 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. These contents is mainly owed to [39],[53]. Next, we explain the basic concepts of automata and formal languages. After then, we introduce four types of prefix codes generated by Petri nets and their fundamental properties[47].

2.1 Petri net

We introduce the definition of a Petri net and its related concepts and notation.

2.1.1 Definitions and Notation

A Petri net is viewed as a particular kind of directed graph, together with an initial state μ_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 μ_0 is denoted by (N, μ_0). □

Remark A Petri net is often given as a 5-tuple (P, T, F, W, μ_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 μ, an n-dimensional row vector, where n is the total number of places. The p-th conponent of μ, 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 μ_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 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 μ 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

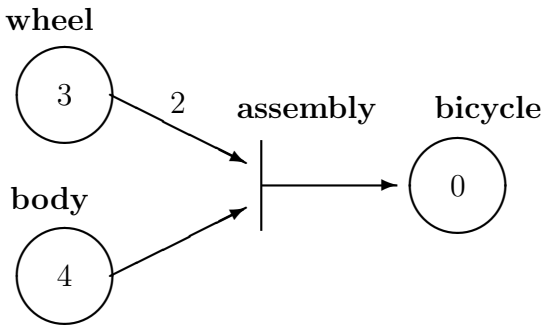

Figure 2.1: Graphical representation of a Petri net

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 μ is replaced with the following new marking μ':

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

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 $i\,(1 \le i \le n)$. Then we often also use the notation $\mu\ [\sigma\rangle\ \mu'$. In particular, a firing sequence σ is said to be *positive* if all $\mu_i\,(1 \le i \le n)$ are positive.
(4) A marking μ is said to be *reachable* from the initial marking μ_0 if there exists a firing sequence σ such that $\mu_0\ [\sigma\rangle\ \mu$. Then μ is said to be reachable from μ_0 through σ. The set of all possible markings reachable from μ_0 in a Petri net (N, μ_0) is denoted by $R(N, \mu_0)$ or simply $R(\mu_0)$. The set of all possible firing sequences from μ_0 in a Petri net (N, μ_0) is denoted by $L(N, \mu_0)$ or simply $L(\mu_0)$. The set of all possible positive firing sequences from μ_0 in (N, μ_0) is denoted by $L_+(N, \mu_0)$ or simply $L_+(\mu_0)$.

The *transition function* (or *next-state function*) δ_{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 δ_{PN} by δ 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 μ_0 to $\mu_1 = (3, 1, 1)$. Then **assembly** is not enabled under μ_1 because $W(\mathbf{wheel}, \mathbf{assembly}) = 2 \leq \mu(\mathbf{wheel}) = 1$ 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 δ of PN, $\delta(\mu_0, 1) = (4, 3, 0)$, $\delta(\mu_0, \mathbf{assembly}) = (3, 1, 1)$, $\delta(\mu_0, \mathbf{assembly\ assembly})$ is undefined. □

For ease of expression, the following notations will be used extensively throughout the literature. Let (P, T, W, μ) be a Petri net, $p \in P$ be a place, $t \in T$ be a transition and σ 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.

> $\Delta(t)$ 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 σ is a firing sequence of the Petri net.

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.

2.2 Languages and Codes

We explain terms and notations related to the formal language theory in Section 2.2.1, and codes in Section 2.2.2

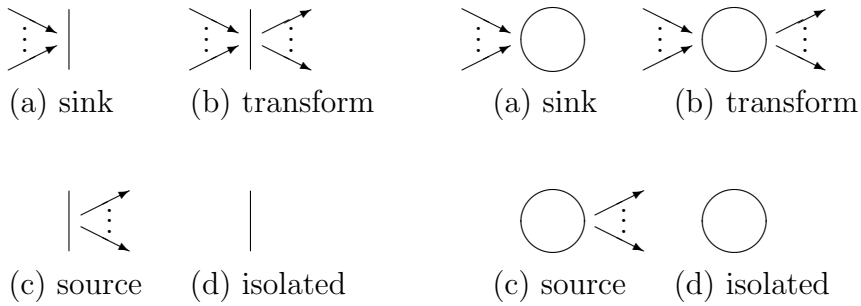

Figure 2.2: Classification of transitions

2.2.1 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 $|x| = 1 \iff x \in X$.

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

$$(a_1 a_2 \dots a_m) \cdot (b_1 b_2 \dots b_n) = a_1 a_2 \dots a_m b_1 b_2 \dots b_n,$$

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

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

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.

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 $x = x_1 u x_2, y = x_1 v x_2$, 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:

context-sensitive(or type 1) if each $u \to v \in P$ has $u = u_1 A u_2, v = u_1 x u_2$ 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 production $u \to v \in P$ has $u \in N$ and $v \in X^* N X^*$.

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

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

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

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: $REG \subset LIN \subset CF \subset CS \subset RE$.

2.2.2 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, $v_1uv_2 \in L$ 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].

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

DEFINITION 2.3.1 Let $PN = (N, \mu_0) = (P, X, W, \mu_0)$ be a Petri net. The set

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

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

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

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 2.3.2 Let $PN = (N, \mu_0) = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_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 2.3.1 Let $PN = (P, T, W, \mu_0)$ be a Petri net where $\mu_0 = (2, 4)$ shown in Figure 2.3. Observing the reachability graph of PN, we obviously have $\text{Stab}(\mu_0) = \{a^3b, a^2ba\}^*$ and $\mathbb{S}(PN) = \{a^3b, a^2ba\}$. An element in $\mathbb{S}(PN)$ is a nonempty minimal word in $\text{Stab}(\mu_0)$ with respect to the prefix order $\leq_p$. Since an element in $\mathbb{D}(PN)$ is a positive firing sequence σ in $\mathbb{S}(PN)$ which for any prefix u of σ $\delta(\mu_0, u)$ must be positive, a^3b is in $\mathbb{D}(PN)$ but a^2ba is not. Hence, $\mathbb{D}(PN) = \{a^3b\}$. □

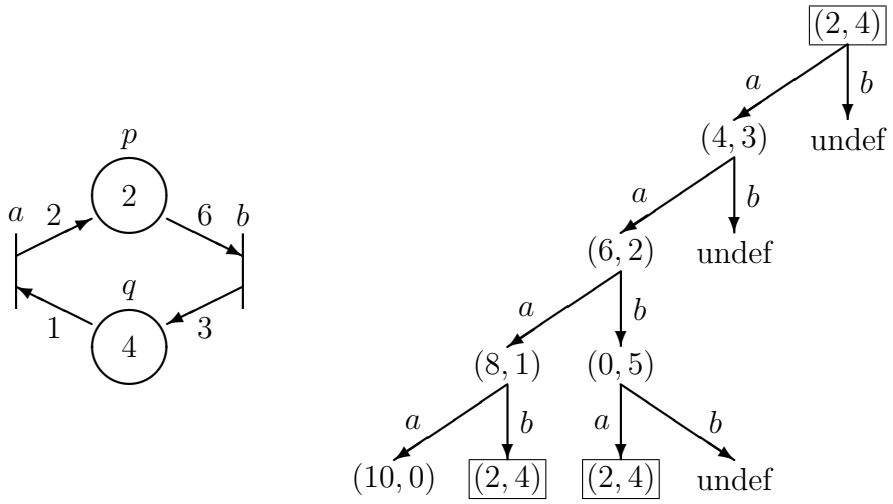

Figure 2.3: Petri net and its related codes

DEFINITION 2.3.3 Let $PN = (N, \mu_0) = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_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:

(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 2.3.4 Let $PN = (N, \mu_0) = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_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 2.3.3, $\mathbb{C}(PN)$ 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 2.3.1 [47]Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_0. Then

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

□

EXAMPLE 2.3.2 Again let $PN = (P, T, W, \mu_0)$ be a Petri net where $\mu_0 = (2, 4)$ shown in Figure 2.3. $\mathbb{C}(PN, \mu_0) = \{a^3b, a^2ba\}^*\{a^4, a^2b\}$. 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 δ 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.

The notion of maximality of a CPN code is very important in relation to liveness in the following sense. Let $\mathbb{C}(PN) \neq \emptyset, PN = (P, X, W, \mu_0)$ with a positive marking μ_0 be a maximal CPN code. If $\mu \in L_+(\mu_0)$, that is, μ is reachable from μ_0 by a positive firing sequence, then every transition is enable at the marking μ. In other words, the assumption is equivalent to $u \in \mathbb{C}(PN)(X^+)^{-1}$, that is, u is a proper prefix of $\mathbb{C}(PN)$.

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 $F = \{(p,t) \in P \times T \,|\, W(p,t) > 0\} \cup \{(t,p) \in T \times P \,|\, W(t,p) > 0\}$. 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

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

□

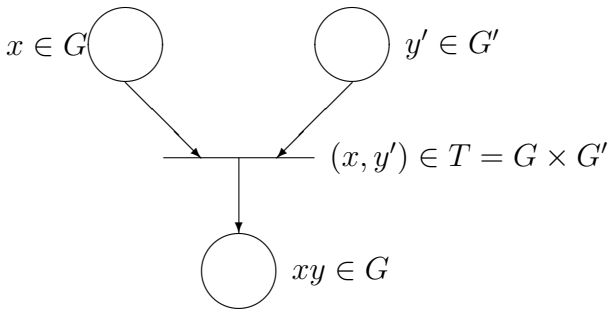

Figure 3.1: Structure of the Transformation Net of G

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. □

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 $\mathbf{Aut}(G)$. □

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 μ and λ be markings with $\lambda(g') = 0$ for any $g' \in G'$ (we may assume this condition without loss of generality). Then if μ is reachable from λ, 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 compute and check the conditions (1) and (2) in $O(n)$ time, where $n = |G|$.

The remaining problem is the reverse implication, that is, whether both (1) and (2) imply that μ is reachable from λ 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 μ_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 μ_0.

Let δ 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 $u = a_1 a_2 \ldots a_r \in X^*$ 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 $n, n \geq 1$. □

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. □

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. □

COROLLARY 4.1.2 The property being a maximal CPN code over X is not preserved under concatenation. □

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 $a^{min\{|u||u\in C\}} \in C$. □

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. □

4.2 Maximal CPN Codes of the Form AB

We can generalize PROPOSITION 4.1.1 to a maximal CPN code of the form AB 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. □

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 $(\cup_{b\in X} \{(P_r(A)\backslash A)\diamond Bb^{-1}\}b) \cup (\cup_{a\in X}\{(P_r(B)\backslash B)\diamond Aa^{-1}\}a)$ 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. □

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 $A = B = X$. □

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 λ-free homomorphism h of Y^* onto X^* such that $A \oplus B = h(D)$. □

THEOREM 4.3.1 The property being a maximal CPN code over X is not preserved under λ-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 $baba \in C$. □

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 $m(A \cup B)$. □

PROPOSITION 4.3.4 Let A, B be maximal CPN codes over X. Then, $A \otimes B$ is a maximal CPN code over X. □

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 PROPOSITION 4.3.4 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 $n = max\{|u| \mid u \in A\}$. 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, PROPOSITION 4.3.4 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 $A \subseteq B$. □

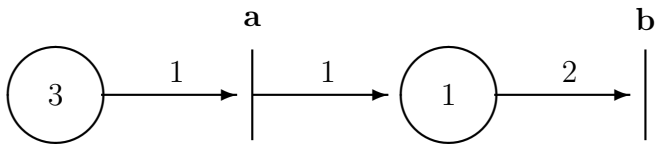

Figure 4.1: Petri net D with $\mathbb{C}(D) = \{ab, aaba, aaa\}$

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 μ_0 such that $\mathbb{C}(D) = A$, and δ 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 δ' 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 $r(A) = 1$. □

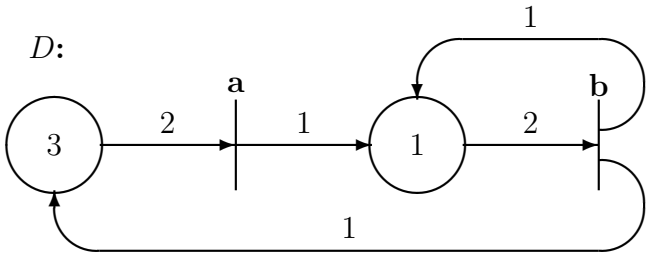

Figure 4.2: Petri net D with $r(\mathcal{L}(D)) > 1$

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 ?*

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. □

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. □

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. □

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**. □

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 codes.

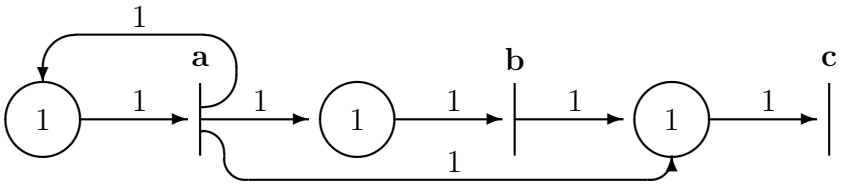

Figure 4.3: Petri net which generates a non-context-free language

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

Before giving the second proof, we provide a few notations. Let $\mu_1, \mu_2, \ldots, \mu_r$ and μ 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 μ_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 Ω_D we denote the set of markings $\{\mu \,|\, \forall p \in P, \mu(p) \leq M_D + 3N_D\}$. Notice that Ω_D is a finite set.

(Sketch of Proof) Let $D = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_0. δ 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

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

$R_8 = \{[w][\pi_p] \rightarrow w \,|\, p \in P, w \in X^2 \cup X^3\}$

Consequently, $\mathcal{L}(G) = \mathbb{C}(D)$ □

Chapter 5

Maximality of CPN Codes

In Chapter 5 we treat the open problem 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 later 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 ≤ 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 μ_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 μ_0, K be its critical place mapping. If $C = \mathbb{C}(PN)$ is a

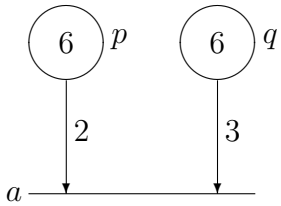

Figure 5.1: Example of a critical place, $K(a) = \{q\}$.

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 $W(p,a) = W(p,b)$.

□

THEOREM 5.1.2 (Deletion of useless places) Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_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 $(P' \times X) \cup (X \times P')$,
μ_0' is a restriction of μ_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. □

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 δ is the next-state function of (P, X, W, μ_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 μ_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

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

□

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.

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

□

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.

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 μ_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. □

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 μ_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, μ_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 $n_q \not| W(q, a)$ or $n_q \not| W(a, q)$, nor $b \in X_q$ such that $n_p \not| W(p, b)$ or $n_p \not| W(a, b)$, 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 $PN' = (P', X', W', \mu_0')$:

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

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:

$$
\begin{array}{l}
\text{(A)}\ \exists a \in X_p,\ \exists b \in X_q\,[x = W(p,b) > 0, y = W(q,a) > 0, \text{ and } (n_p \not| x \text{ or } n_q \not| y)] \\
\text{(B)}\ \exists a \in X_p,\ \exists b \in X_q\,[x = W(b,p) > 0, y = W(q,a) > 0, \text{ and } (n_p \not| x \text{ or } n_q \not| y)] \\
\text{(C)}\ \exists a \in X_p,\ \exists b \in X_q\,[W(b,p) = 0, y = W(q,a) > 0, \text{ and } n_q \not| y] \\
\text{(D)}\ \exists a \in X_p,\ \exists b \in X_q\,[x = W(b,p) > 0, y = W(a,q) > 0, \text{ and } (n_p \not| x \text{ or } n_q \not| y)] \\
\text{(E)}\ \exists a \in X_p,\ \exists b \in X_q\,[x = W(b,p) > 0, W(q,a) = 0, \text{ and } n_p \not| x]
\end{array}
$$

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 μ_0 which is satisfied the condition (A) in the proof of THEOREM 5.2.1. 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. □

LEMMA 5.2.2 Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_0 which is satisfied the condition (B) in the proof of THEOREM 5.2.1. If $C = \mathbb{C}(PN) \neq \emptyset$ is an maximal CPN code, then it is an input-ordinary CPN code. □

LEMMA 5.2.3 Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_0 which is satisfied the condition (C) in the proof of THEOREM 5.2.1. If $C = \mathbb{C}(PN) \neq \emptyset$ is a maximal CPN code,then it is an input-ordinary CPN code. □

LEMMA 5.2.4 Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_0 which is satisfied the condition (D) in the proof of THEOREM 5.2.1. If $C = \mathbb{C}(PN)$ cannot be a maximal CPN code. □

LEMMA 5.2.5 Let $PN = (P, X, W, \mu_0)$ be a Petri net with a positive marking μ_0 which is satisfied the condition (E) in the proof of THEOREM 5.2.1. If $C = \mathbb{C}(PN) \neq \emptyset$ is a maximal CPN code,then it is an input-ordinary CPN code. □

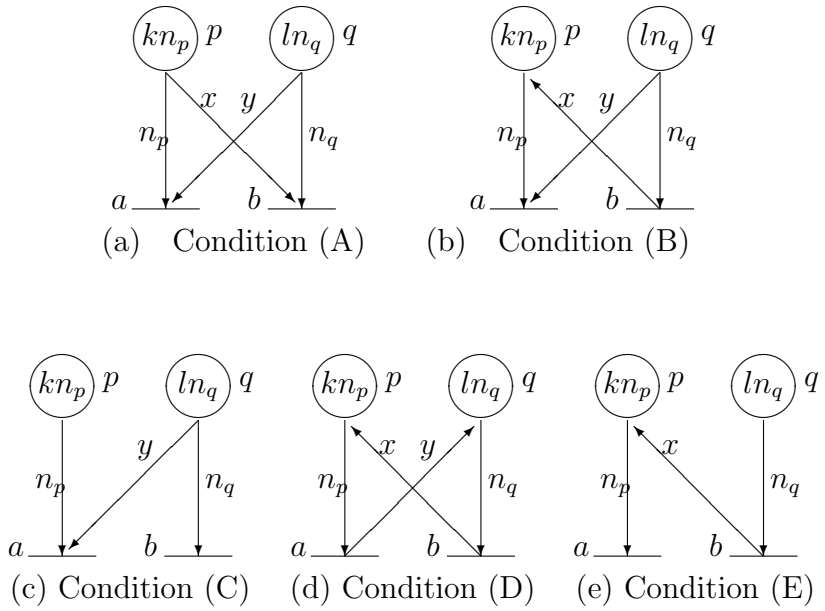

Figure 5.2: Arc connection under conditions (A) to (E).

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 $\mu_0(p) - d \times i \geq 0$, there is a word $w \in X^+$ such that

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

□

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 μ_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) $n_p | \mu_0(p)$. □

COROLLARY 5.2.1 Let $PN = (P, X, W, \mu_0)$ be a Petri net with at least one source transitions and μ_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, μ_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. □

LEMMA 5.2.9 Let $PN = (P, X, W, \mu_0)$ be a Petri net with source transitions, μ_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. □

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, μ_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, μ_0 be positive and $|P| = 2$. If $C = \mathbb{C}(PN)$ is a maximal CPN code, then C is an input-ordinary CPN code. □

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, μ_0) is the set of all nonpositive firing sequences in (N, μ_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 λ-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 μ_0 through a positive firing sequence in (N, μ_0). This condition is obviously true if (N, μ_0) is input-ordinary. Conversely we wonder whether the set of all positive firing sequences in a general Petri net (N, μ_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, μ_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 ≤ 2, and in the case that the number of transitions is equal to 1. It still remains open in a general Petri net.

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.