Earley's Parsing Algorithm and k-Petri Net Controlled Grammars

Taishin Y. Nishida

Department of Information Sciences, Toyama Prefectural University,
Imizu, 939-0398 Toyama, Japan
nishida@pu-toyama.ac.jp

Abstract. In this paper we modify Earley's parsing algorithm to parse words generated by Petri net controlled grammars. Adding a vector which corresponds to a marking of a Petri net to Earley's algorithm, it is shown that languages generated by a subclass of k-Petri net controlled grammars (introduced by J. Dassow and S. Turaev) are parsed in polynomial time of the length of a word.

1 Introduction

Petri net controlled grammars have been introduced by M. ter Beek and J. Kleijn [1] and then have been extensively studied by J. Dassow and S. Turaev [3–6, 13]. For a context-free grammar, there is a Petri net whose places correspond to the nonterminals of the grammar and whose transitions correspond to the rules of the grammar such that a transition occurs (fires) if and only if the corresponding rule is applied in a derivation of the grammar. That is, the Petri net, called a cf Petri net, represents a sequence of rules which are used in a derivation of the grammar. Adding new places to a cf Petri net, a Petri net controls derivation of a context-free grammar to generate a non-context-free language. Thus Petri net controlled grammars have appeared quite naturally.

In this paper we focus our attention on the membership problem of Petri net controlled grammars. It has been shown that languages generated by most variants of Petri net controlled grammars are included in the class of matrix languages [3, 4, 13]. Thus the membership problem of Petri net controlled grammars might be reduced to that of matrix grammars. But we want to solve the membership problem directly, that is, to parse a word in order to construct a derivation tree. If a variant of Petri net controlled grammars has a fast parsing algorithm, then the grammars will be as easily and frequently used as context-free (without control) grammars, or even will replace context-free grammars, in practical application.

There are two famous fast parsing algorithms, CKY algorithm [10, 14] and Earley's algorithm [8], for context-free grammars. CKY algorithm assumes a grammar in Chomsky normal form. But, since modification of production rules in a Petri net controlled grammar changes the structure of the Petri net, it is not clear that a language generated by a variant of Petri net controlled grammars

H. Bordihn, M. Kutrib, and B. Truthe (Eds.): Dassow Festschrift 2012, LNCS 7300, pp. 174–185, 2012.

is generated by a grammar in the same variant and in Chomsky normal form. This is why CKY algorithm is inappropriate for our purpose. On the other hand, Earley's algorithm has no restriction on context-free grammars. Thus we start to develop a parsing algorithm for Petri net controlled grammars from Earley's algorithm.

In Section 2 basic notions and notations about context-free grammars, Petri nets, and Earley's algorithm are described. A variant of Petri net controlled grammars, k-Petri net controlled grammars (k-PN controlled grammars for short), is introduced in Section 3. Earley's algorithm is extended in Section 4 to parse words generated by k-PN controlled grammars. The algorithm parses a word of length n in time $O(n^3)$ for an unambiguous k-PN controlled grammar and in time $O(n^{k+4})$ for an ambiguous grammar. Section 5 is a conclusion.

2 Preliminaries

We assume that the reader is familiar with rudiments of context-free grammars, regulated grammars, and Petri nets. For notions and notations which are not described in this section, we refer to [2, 7, 9, 11, 12].

2.1 Context-Free Grammars

A context-free grammar is a construct $G = (V, \Sigma, S, R)$ where V and Σ are *nonterminal* and *terminal* alphabets, respectively, with $V \cap \Sigma = \emptyset$, $S \in V$ is the *start symbol*, and $R \subseteq V \times (V \cup \Sigma)^*$ is a finite set of (*production*) *rules*. A rule (A, x) is written as $A \to x$. A word $x \in (V \cup \Sigma)^+$ directly derives $y \in (V \cup \Sigma)^*$, written as $x \overset{r}{\Rightarrow}_G y$, if and only if there is a rule $r : A \to \alpha \in R$ such that $x = x_1 A x_2$ and $y = x_1 \alpha x_2$. We write $x \overset{r}{\Rightarrow} y$ if G is understood and write $x \Rightarrow y$ if we are not interested in the rule r. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. If there are a sequence of rules $r_1, r_2, \ldots, r_n$ and a sequence of words $w_0, w_1, \ldots, w_n$ such that $w_{i-1} \overset{r_i}{\Rightarrow} w_i$ for every $1 \leq i \leq n$, then we write $w_0 \overset{r_1 r_2 \cdots r_n}{\Longrightarrow} w_n$. We call the sequence of rules $r_1, r_2, \ldots, r_n$ *derivation process* from w_0 to w_n. The language generated by G is defined by $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

Let $G = (V, \Sigma, S, R)$ be a context-free grammar. A rule of the form $A \to \lambda$ is called a λ-*rule*, where λ is the empty word. A rule $A \to \alpha$ is said to be a *chain rule* if $\alpha \in V$. Let σ be a derivation process from S to $w \in \Sigma^*$ in G. Then σ determines a derivation tree, which is denoted by $t(\sigma)$. Let U be the set of nodes of $t(\sigma)$ and let $\nu : U \to V \cup \Sigma \cup \{\lambda\}$ be the labelling function of $t(\sigma)$. We make a *derivation tree in rules*, denoted by $t_R(\sigma)$, by:

(1) Removing every node u which satisfies $\nu(u) \in \Sigma \cup \{\lambda\}$, i.e., a node which is labelled by a terminal or λ.
(2) Replacing every label of nonterminal with the rule which rewrites the nonterminal in the derivation process σ. That is, define a new labelling function $\nu' : U \to R$ by $\nu'(u) = r$ where rule r rewrites the nonterminal $\nu(u)$.

2.2 Petri Net

A Petri net is a quadruple $N = (P, T, F, \phi)$ where P and T are disjoint finite sets of places and transitions, respectively, $F \subseteq (P \times T) \cup (T \times P)$ is the set of directed arcs, $\phi : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a weight function with $\phi(x, y) = 0$ for every $(x, y) \notin F$, where $\mathbb{N}$ is the set of nonnegative integers. A Petri net can be represented by a bipartite directed graph with the node set $P \cup T$ where places are drawn as circles, transitions are rectangles, and arcs as arrows with labels $\phi(p, t)$ or $\phi(t, p)$. If $\phi(p, t) = 1$ or $\phi(t, p) = 1$, then the label is omitted.

A place contains a number of tokens. Each number of tokens in every place is expressed by a mapping $\mu : P \to \mathbb{N}$, which is called a *marking*. For every place $p \in P$, $\mu(p)$ denotes the number of tokens in p. Graphically, tokens are drawn as small solid dots inside circles.

A transition $t \in T$ is enabled by a marking μ if and only if $\mu(p) \geq \phi(p, t)$ for every $p \in P$. In this case t can *occur* (*fire*). An occurrence of a transition t transforms the marking μ into a new marking μ' which is defined by $\mu'(p) = \mu(p) - \phi(p, t) + \phi(t, p)$ for every $p \in P$. More than one transition may be enabled by a marking. In this case one transition is nondeterministically selected and fires. If a transition t occurs in a marking μ and the marking changes to μ', then we write $\mu \xrightarrow{t} \mu'$. A finite sequence $t_1 t_2 \cdots t_k$ of transitions is called an *occurrence sequence* enabled at a marking μ if there are markings $\mu_1, \mu_2, \ldots, \mu_k$ such that $\mu \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \cdots \xrightarrow{t_k} \mu_k$. In short this sequence can be written as $\mu \xrightarrow{t_1 t_2 \cdots t_k} \mu_k$ or $\mu \xrightarrow{\nu} \mu_k$ where $\nu = t_1 t_2 \cdots t_k$. For each $1 \leq i \leq k$, the marking μ_i is called *reachable* from the marking μ.

A *marked* Petri net is a system $N = (P, T, F, \phi, \iota)$ where (P, T, F, ϕ) is a Petri net, ι is the *initial marking*. Let M be a set of marking. An occurrence sequence ν of transitions is called *successful* for M if it is enabled at the initial marking ι and finished at a final marking τ of M. Thus M is called a set of final markings. If M is understood from the context, we say that ν is a successful occurrence sequence.

Let $N = (P, T, F, \phi)$ be a Petri net. For an arc $e = (u, v)$ in F (note that ($u \in P$ and $v \in T$) or ($u \in T$ and $v \in P$)), we use the notations ${}^\bullet e = u$ and $e^\bullet = v$. A sequence of arcs $e_1, e_2, \ldots, e_n$ is said to be a *path* in N if $e_i^\bullet = {}^\bullet e_{i+1}$ for every $i \in \{1, \ldots, n-1\}$. A path is a *cycle* if $e_n^\bullet = {}^\bullet e_1$.

2.3 Earley's Algorithm

Here we introduce Earley's algorithm [8]. Let $G = (V, \Sigma, S, R)$ be a context-free grammar. Let $G' = (V \cup \{S'\}, \Sigma, S', R' = R \cup \{S' \to S\})$ be a new context-free grammar where S' is a new nonterminal. For a rule $r : A \to \alpha$ in P', an item $[A \to \beta \cdot \gamma]$ is said to be an *Earley's state* where $\beta\gamma = \alpha$. Let $S' \Rightarrow^* w = a_1 a_2 \cdots a_m \in \Sigma^*$ be a derivation in G'. If a rule $A \to \alpha\beta$ which is used in the derivation

$$S' \Rightarrow^* \gamma A \delta \Rightarrow \gamma\alpha\beta\delta \Rightarrow^* a_1 \cdots a_k a_{k+1} \cdots a_i \beta\delta$$

satisfies

$$\gamma \Rightarrow^* a_1 \cdots a_k \quad \text{and} \quad \alpha \Rightarrow^* a_{k+1} \cdots a_i,$$

then the state $[A \to \alpha \cdot \beta]$ belongs to a set $E_w(k,i)$. The sets $E_w(k,i)$ of such states $(0 \leq k \leq i \leq m)$ are called *sets of Earley's state*. The next property directly follows from the definition.

Property 1. Let G, G', and w be grammars and a word described in the above paragraph.

(1) If a state of the form $[A \to \cdot\alpha]$ appears in $E_w(k,i)$, then $k = i$. Thus $[S' \to \cdot S] \in E_w(0,0)$.
(2) $[A \to \alpha\cdot]$ is in $E_w(k,i)$ if and only if $A \Rightarrow^* a_{k+1} \cdots a_i$.
(3) $[S' \to S\cdot]$ is in $E_w(0,m)$ if and only if $S' \Rightarrow^* a_1 \cdots a_m$, that is, w is in $L(G)$.
(4) If $[A \to \alpha \cdot B\beta] \in E_w(k,i)$ with $B \in V$, then $[B \to \cdot\gamma] \in E_w(i,i)$ for every B-rule $B \to \gamma$.
(5) If $[A \to \alpha \cdot B\beta] \in E_w(k,i)$ and $[B \to \gamma\cdot] \in E_w(i,j)$, then $[A \to \alpha B \cdot \beta] \in E_w(k,j)$.
(6) If $[A \to \alpha \cdot a\beta] \in E_w(k,i-1)$ and $a_i = a$, then $[A \to \alpha a \cdot \beta] \in E_w(k,i)$.

The above property shows that an algorithm which constructs every set of Earley's states solves the membership problem for context-free languages. Algorithm 1 constructs sets of Earley's states.

Algorithm 1. An algorithm which constructs sets of Earley's states

input: a word $w = X_1X_2 \cdots X_m$ $(X_i \in \Sigma)$
output: the sets of Earley's states $E_w(k,i)$ $(0 \leq k \leq i \leq m)$
1: for every k, i with $((k,i) \neq (0,0))$
2: $E_w(k,i) = \emptyset$
3: $E_w(0,0) = \{[S' \to \cdot S]\}$
4: do
5: if $[A \to \alpha \cdot B\beta] \in E_w(0,0)$ and $[B \to \gamma\cdot] \in E_w(0,0)$, then
6: insert $[A \to \alpha B \cdot \beta]$ to $E_w(0,0)$
7: if $[A \to \alpha \cdot B\beta] \in E_w(0,0)$ with $B \in V$, then
8: insert $[B \to \cdot\gamma]$ to $E_w(0,0)$ for every B-rule $B \to \gamma$
9: while $E_w(0,0)$ is changed
10: do
11: if $[A \to \alpha \cdot a\beta] \in E_w(k,i-1)$ and $X_i = a$, then
12: insert $[A \to \alpha a \cdot \beta]$ in $E_w(k,i)$
13: if $[A \to \alpha \cdot B\beta] \in E_w(k,i)$ and $[B \to \gamma\cdot] \in E_w(i,j)$, then
14: insert $[A \to \alpha B \cdot \beta]$ to $E_w(k,j)$
15: if $[A \to \alpha \cdot B\beta] \in E_w(k,i)$ with $B \in V$, then
16: insert $[B \to \cdot\gamma]$ to $E_w(i,i)$ for every B-rule $B \to \gamma$
17: while some $E_w(k,i)$ is changed

For an Earley's state $[A \to \alpha\cdot] \in E_w(k,i)$ a set of derivations from A to $a_k \cdots a_i$ can be constructed inductively.

(1) If $\alpha \in \Sigma^*$, then $A \Rightarrow \alpha$ is a derivation.

(2) If $\alpha = u_0B_1u_1 \cdots u_{l-1}B_lu_l$ for some $B_1, \ldots, B_l \in V$ and $u_0u_1 \cdots u_l \in \Sigma^*$ and a set of derivations from B_i to β_i has been constructed for every $i \in \{1, \ldots, l\}$, then the set of derivations from A to $a_k \cdots a_i$ contains all derivations of the form

$$A \Rightarrow u_0B_1u_1 \cdots u_{l-1}B_lu_l \Rightarrow^* u_0\beta_1u_1 \cdots u_{l-1}B_lu_l \Rightarrow^* \cdots$$

$$\Rightarrow^* u_0\beta_1u_1 \cdots u_{l-1}\beta_lu_l = a_k \cdots a_i.$$

If a start symbol S of a grammar does not appear in a right-hand side of any rule, then we do not introduce new start symbol S' and we use states of the form $[S \to \alpha\cdot]$ to decide whether a word is generated or not. Whenever we discuss Earley's states, a new start symbol S' is implicitly assumed if necessarily.

3 *k*-Petri Net Controlled Grammars

In this section we define k-Petri net controlled grammars according to [6].

Let $G = (V, \Sigma, S, R)$ be a context-free grammar. A marked Petri net $N = (P, T, F, \phi, \iota)$ is a *cf Petri net* with respect to G under labelling functions (β, γ) if N and (β, γ) satisfy:

(1) $\beta : P \to V$ and $\gamma : T \to R$ are bijections.
(2) F and ϕ satisfy:
- $(p, t) \in F$ if and only if $\gamma(t) = A \to \alpha$ and $\beta(p) = A$, in this case $\phi(p, t) = 1$.
- $(t, p) \in F$ if and only if $\gamma(t) = A \to \alpha$ and $\beta(p) = x$ where $|\alpha|_x \geq 1$, in this case $\phi(t, p) = |\alpha|_x$.

(3) $\iota(p) = 1$ if $\beta(p) = S$ and $\iota(p) = 0$ for every $p \in P - \{\beta^{-1}(S)\}$.

We note that a cf Petri net is uniquely determined from a combination of a context-free grammar G and a pair of labelling functions (β, γ). Therefore, a cf Petri net with respect to G under (β, γ) can be denoted by $PN[G, (\beta, \gamma)]$.

Definition 1. *Let $G_0 = (V, \Sigma, S, R)$ be a context-free grammar and let $N = PN[G_0, (\beta, \gamma)] = (P, T, F, \phi, \iota)$ be a cf Petri net with respect to G_0. A* k-Petri net controlled grammar *(k-*PN controlled grammar*) is a quintuple $G = (V, \Sigma, S, R, N_k)$ where V, Σ, S, R are the components from the grammar G_0 and $N_k = (P', T', F', \phi', \iota')$ is a k-Petri net which satisfies:*

(1) $P' = P \cup Q$ where $Q = \{q_1, \ldots, q_k\}$ is a set of new places.
(2) $T' = T$.
(3) $F' = F \cup E$ where $E \subseteq (T \times Q) \cup (Q \times T)$ is a set of new arcs. Every arc in E satisfies the following condition;
- *for every $t \in T$, $(t, q_i) \in E$ and $(t, q_j) \in E$ imply $i = j$ and $(q_i, t) \in E$ and $(q_j, t) \in E$ imply $i = j$.*
- *for every $1 \leq i < j \leq k$, there exists no $t \in T$ such that $(t, q_i) \in E$ and $(q_j, t) \in E$.*

- *for every* $1 \leq i \leq k$, $(q_i, t) \in E$ *for some* $t \in T$ *if and only if* $(t', q_i) \in E$ *for some* $t' \in T$.

(4) $\phi'(x, y) = \phi(x, y)$ *if* $(x, y) \in F$ *and* $\phi'(x, y) = 1$ *if* $(x, y) \in E$.
(5) $\iota'(p) = 1$ *if* $\beta(p) = S$ *and* $\iota'(p) = 0$ *for every* $p \in (P - \{\beta^{-1}(S)\}) \cup Q$, *i,e.,* $\iota'(p) = \iota(p)$ *if* $p \in P$ *and* $\iota'(p) = 0$ *if* $p \in Q$.

We call G_0 *the* underlying grammar *of* G.

In [6], condition (3) is described differently:

– $E = \{(t, q_i) \mid t \in T_1^i, 1 \leq i \leq k\} \cup \{(q_i, t) \mid t \in T_2^i, 1 \leq i \leq k\}$ such that $T_1^i \subset T$ and $T_2^i \subset T$, $1 \leq i \leq k$ where $T_l^i \cap T_l^j = \emptyset$ for $1 \leq l \leq 2$, $T_1^i \cap T_2^j = \emptyset$ for $1 \leq i < j \leq k$ and $T_1^i = \emptyset$ if and only if $T_2^i = \emptyset$ for any $1 \leq i \leq k$.

It is clear that the two conditions say the same thing. The most important point of condition (3) is that a k-Petri net does not have any cycle of arcs in E. We call it *cycle-free* condition.

Let τ be the marking $\tau(p) = 0$ for every $p \in P \cup Q$. Next we define the derivation in a k-PN controlled grammar G and the language generated by G.

Definition 2. *Let* $G = (V, \Sigma, S, R, N_k)$ *be a* k*-PN controlled grammar. A word* $\alpha \in (V \cup \Sigma)^*$ *is* derived *in* G *if* $S \stackrel{r_1 r_2 \cdots r_n}{\Longrightarrow} \alpha$ *such that* $t_1 t_2 \cdots t_n = \gamma^{-1}(r_1 r_2 \cdots r_n) \in T^*$ *is an occurrence sequence of the transitions of* N_k *enabled at the initial marking* ι. *A derivation* $S \stackrel{r_1 r_2 \cdots r_n}{\Longrightarrow} w \in \Sigma^*$ successfully generates *a terminal word if* $t_1 t_2 \cdots t_n = \gamma^{-1}(r_1 r_2 \cdots r_n) \in T^*$ *is an occurrence sequence of the transitions of* N_k *enabled at the initial marking* ι *and finished at the final marking* τ. *The* language generated *by* G, *denoted by* $L(G)$, *consists of all words which are successfully generated in* G[1].

In k-PN controlled grammars, new places control sequence of rules in a derivation, which is shown in the next example.

Example 1 (Example 7 of [3]). Let $G = (\{S, A, B\}, \{a, b, c\}, S, R, N_1)$ be a 1-PN controlled grammar where R consists of the following rules

$$r_0 : S \to AB,\ r_1 : A \to aAb,\ r_2 : A \to ab,\ r_3 : B \to cB,\ r_4 : B \to c$$

[1] A k-PN controlled grammar can be viewed a kind of *positive* valence grammar [6]. A (context-free) valence grammar G on $\mathbf{Z}^k$ for some positive integer k is a construct $G = (V, \Sigma, S, R, \nu)$ where $\mathbf{Z}$ is the set of integers, $(V, \Sigma, S, R) = G_0$ is a context-free grammar (the underlying grammar), and $\nu : R \to \mathbf{Z}^k$ is the valence function. Let $\sigma = (r_1, r_2, \ldots, r_n)$ be a derivation process from S to $w \in \Sigma^*$ in G_0. The word w is generated by G if and only if $\sum_{j=1}^{n} \nu(r_j) = \mathbf{0}$ and for every initial segment $(r_1, r_2, \ldots, r_i)$ of σ $\sum_{j=1}^{i} \nu(r_j) \geq \mathbf{0}$ where $\boldsymbol{a} \geq \boldsymbol{b}$ if and only if $a_j \geq b_j$ for every jth component. A k-PN controlled grammar (V, Σ, S, R, N_k) is a positive valence grammar (V, Σ, S, R, ν) where $\nu : R \to \mathbf{Z}^k$ is given by

$$\nu(r) = (\phi(\gamma^{-1}(r), q_1) - \phi(q_1, \gamma^{-1}(r)), \ldots, (\phi(\gamma^{-1}(r), q_k) - \phi(q_k, \gamma^{-1}(r)).$$

Clearly, every component of $\nu(r)$ is one of 1, 0, or -1. It should be noted that the notion of positive valence grammars is different from that of valence grammars. The latter permits $\sum_{j=1}^{i} \nu(r_j) \not\geq \mathbf{0}$ for some initial segment $(r_1, \cdots, r_i)$.

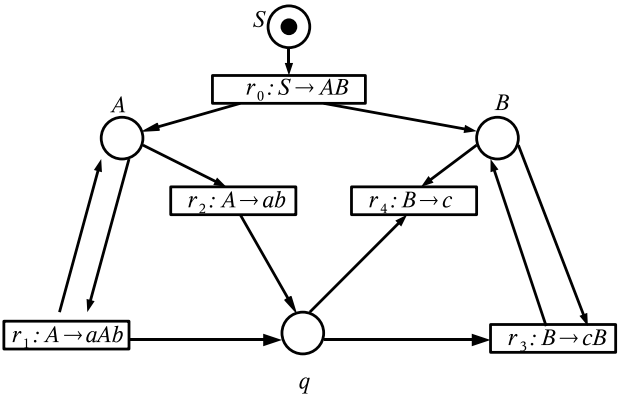

Fig. 1. 1-Petri net controlled grammar generating $\{a^n b^n c^n \mid n > 0\}$

and N_1 is illustrated in Fig. 1, in which rules are drawn in the rectangles of the corresponding transitions. The grammar G generates the language $\{a^n b^n c^n \mid n > 0\}$. □

4 Parsing for *k*-Petri Net Controlled Grammars

Our aim of this section is to develop a parsing algorithm for k-PN controlled grammars.

First we must consider chain rules and λ-rules. Every context-free grammar can be converted, without changing the generated language, to a grammar with no chain rules and no λ-rules. But it is not obvious whether a k-PN controlled grammar has an equivalent grammar (generating the same language) with no chain rules and no λ-rules since a chain rule or a λ-rule may make a token in Q. Indeed these rules cause a subtle problem (see Section 5). On the other hand, a grammar with chain rules or λ-rules may have infinite derivation trees for a word, which makes parsing very complex. So, in the remaining sections in this paper, we assume that an underlying context-free grammar does not have any chain rules nor λ-rules (but $S \to \lambda$ for the start symbol S with the condition that S does not appear in a right-hand side of any rule).

A parsing algorithm for k-PN controlled grammars is obtained by associating a vector which corresponds to a marking of a k-Petri net to each Earley's state. Let $G = (V, \Sigma, S, R, N_k)$ be a k-PN controlled grammar and let $G_0 = (V, \Sigma, S, R)$ be the underlying grammar of G. For a word $w \in L(G_0)$ we define an Earley's state with a *token counter*, which is a vector from $\mathbb{Z}^k$ with set of integers $\mathbb{Z}$, as follows:

(1) For a rule $A \to \alpha$, the state $[A \to \cdot\alpha]$ has a token counter $(v_1, \ldots, v_k)$ where $v_i = 1$ and $v_j = 0$ for $j \neq i$ if $(\gamma^{-1}(A \to \alpha), q_i) \in E$, $v_i = -1$ and $v_j = 0$ for $j \neq i$ if $(q_i, \gamma^{-1}(A \to \alpha)) \in E$, or $v_i = 0$ for every $1 \leq i \leq k$ otherwise.

(2) If a state $[A \to \alpha B \cdot \beta]$ is obtained from $[A \to \alpha \cdot B\beta]\boldsymbol{v}_1$ and $[B \to \gamma \cdot]\boldsymbol{v}_2$ where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are token counters associated to the states, then $[A \to \alpha B \cdot \beta]$ has the token counter $\boldsymbol{v}_1 + \boldsymbol{v}_2$ in which the addition is the normal component-wise vector addition.

If the underlying grammar is ambiguous, then there may be states $[A \to \alpha \cdot B\beta] \in E_w(i,j)$, $[B \to \gamma \cdot] \in E_w(j,l)$, $[A \to \alpha \cdot B\beta] \in E_w(i,j')$, and $[B \to \gamma' \cdot] \in E_w(j',l)$. In this case a state $[A \to \alpha B \cdot \beta] \in E_w(i,l)$ is constructed more than one way. The state may have different token counters, that is, the situation $[A \to \alpha B \cdot \beta]\boldsymbol{v} \in E_w(i,l)$ and $[A \to \alpha B \cdot \beta]\boldsymbol{v}' \in E_w(i,l)$ with $\boldsymbol{v} \neq \boldsymbol{v}'$ is possible. States with different token counters should be treated differently.

Example 2. In Example 1, the sets of Earley's states $E_w(k,i)$ with token counters for the word $w = a^2b^2c^2$ are shown in the next table.

	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
	$[S \to \cdot AB](0)$ $[A \to \cdot aAb](0)$ $[A \to \cdot ab](0)$	$[A \to a \cdot Ab](1)$ $[A \to a \cdot b](1)$		$[A \to aA \cdot b](2)$	$[S \to A \cdot B](2)$ $[A \to aAb \cdot](2)$	$[S \to AB \cdot](1)$	$[S \to AB \cdot](0)$
$k=1$		$[A \to \cdot aAb](1)$ $[A \to \cdot ab](1)$	$[A \to a \cdot Ab](1)$ $[A \to a \cdot b](1)$	$[A \to ab \cdot](1)$			
$k=2$			$[A \to \cdot aAb](1)$ $[A \to \cdot ab](1)$				
$k=3$							
$k=4$					$[B \to \cdot cB](-1)$ $[B \to \cdot c](-1)$	$[B \to c \cdot B](-1)$ $[B \to c \cdot](-1)$	$[B \to cB \cdot](-2)$
$k=5$						$[B \to \cdot cB](-1)$ $[B \to \cdot c](-1)$	$[B \to c \cdot B](-1)$ $[B \to c \cdot](-1)$

□

It is obvious that a successful derivation $S \Rightarrow^* w \in \Sigma^*$ in G (under control) implies $[S' \to S \cdot]\mathbf{0} \in E_w(0,n)$ where $\mathbf{0}$ is the zero vector and $n = |w|$. But the converse is not always the case. Let us consider the next example.

Example 3. Let $G = (\{S,A,B,C\}, \{a,b,c\}, S, R, N_1)$ be a 1-PN controlled grammar where R and N_1 are illustrated in Fig. 2.

The word *bbabc* is generated by G while the word *bacc* cannot be generated by G. But, as seen in the next tables, both token counters attached to Earley's states for the words become the zero vector.

Earley's states for the word *bbabc*.

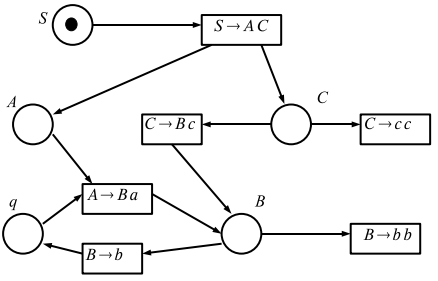

Fig. 2. Petri net of Example 3

$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$[S \to \cdot AC](0)$			$[S \to A \cdot C](-1)$		$[S \to AC\cdot](0)$
$[A \to \cdot Ba](-1)$	$[A \to B \cdot a](0)$	$[A \to B \cdot a](-1)$	$[A \to Ba\cdot](-1)$		
$[B \to \cdot b](1)$	$[B \to b\cdot](1)$				
$[B \to \cdot bb](0)$	$[B \to b \cdot b](0)$	$[B \to bb\cdot](0)$			
$k=1$					
	$k=2$				
			$[C \to \cdot Bc](0)$	$[C \to B \cdot c](1)$	$[C \to Bc\cdot](1)$
		$k=3$	$[C \to \cdot cc](0)$		
			$[B \to \cdot b](1)$	$[B \to b\cdot](1)$	
			$[B \to \cdot bb](0)$	$[B \to b \cdot b](0)$	
			$k=4$		
				$k=5$	

Earley's states for the word *bacc*.

$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
$[S \to \cdot AC](0)$		$[S \to A \cdot C](0)$		$[S \to AC\cdot](0)$
$[A \to \cdot Ba](-1)$	$[A \to B \cdot a](0)$	$[A \to Ba\cdot](0)$		
$[B \to \cdot b](1)$	$[B \to b\cdot](1)$			
$[B \to \cdot bb](0)$	$[B \to b \cdot b](0)$			
$k=1$				
	$k=2$	$[C \to \cdot Bc](0)$		
		$[C \to \cdot cc](0)$	$[C \to c \cdot c](0)$	$[C \to cc\cdot](0)$
		$[B \to \cdot b](1)$		
		$[B \to \cdot bb](0)$		
		$k=3$		
			$k=4$	

□

If a k-Petri net $N_k = (P, T, F, \phi, \iota)$ satisfies "every cycle in N_k does not contain any places in Q", then we can prove equivalence between derivations $S \Rightarrow^* w \in \Sigma^*$ under control of N_k and $[S' \Rightarrow S\cdot]\mathbf{0} \in E_w(0, n)$. We call the condition *strict cycle-free condition*.

Lemma 1. *Let $G = (V, \Sigma, S, R, N_k)$ be a k-PN controlled grammar with strict cycle-free Petri net N_k and let $G_0 = (V, \Sigma, S, R)$ be the underlying grammar of G. For every $w \in L(G_0)$ with $|w| = n$, if $[S' \to S\cdot]\mathbf{0} \in E_w(0, n)$, then $w \in L(G)$.*

Proof. Let $S' \Rightarrow^* w$ be a derivation which is constructed from $[S' \to S\cdot]\mathbf{0} \in E_w(0, n)$. Let $\alpha A \beta$ be a sentential form in the derivation such that there is an arc $(q_l, \gamma^{-1}(A \to \delta)) \in E$ where $A \to \delta$ is the rule in the derivation. If the l-th component of the token counter increases in the derivation $A \Rightarrow \delta \Rightarrow^*$

$u \in \Sigma^*$, then there is a path from $\gamma^{-1}(A \to \delta)$ to q_l in N_k. The path and the arc $(q_l, \gamma^{-1}(A \to \delta))$ forms a cycle, which contradicts to the strict cycle-free condition. Thus the l-th component of the token counter cannot increase in the derivation $A \Rightarrow^* u$.

Since the l-th component of the token counter becomes to 0 after derivation, some positive values in the l-th component must appear in one of the derivations: $S' \Rightarrow^* \alpha A \beta$, $\alpha \Rightarrow^* x$, or $\beta \Rightarrow^* y$ where $xuy = w$. Hence there is a derivation in which an occurrence of a positive value in the l-th component is prior to use the rule $A \to \delta$, that is, $A \to \delta$ can be applied under control of N_k. Therefore, the derivation $S \Rightarrow^* w$ is possible in G. □

Since to associate a token counter to an Earley's state can be done in a constant time, parsing for a strict cycle-free k-PN controlled grammar is performed in time proportional to multiplication of n and the number of Earley's states with token counters. Number of Earley's states for a context-free grammar is $O(n^2)$. Now we enumerate different token counters. At most n rules are used to generate a word of length n because the grammar has no chain rules and no λ-rules. Then at most one token is generated or consumed when a rule is used. For a token counter $\boldsymbol{v} = (v_1, \ldots, v_k)$ sum of all components of the vector $\boldsymbol{v}' = (|v_1|, \ldots, |v_k|)$ is at most n. There are

$$\binom{n+k}{k} = O(n^k)$$

different vectors in $\mathbb{N}^k$ that sum of all components is n. Considering cases that total tokens are less than n, there are $O(n^{k+1})$ different token counters. Therefore time complexity of Earley's algorithm with token counters is $O(n^{k+4})$. We note that if the underlying grammar is unambiguous, then the time complexity is $O(n^3)$.

The strict cycle-free condition, however, is so strict that some k-PN controlled grammars are excluded from the algorithm, an example is illustrated in Example 3. It is an open problem whether every language generated by a k-PN controlled grammar is generated by a strict cycle-free k-PN controlled grammar or not.

5 Conclusion

We have developed a parsing algorithm for k-PN controlled grammars. If an underlying grammar of a k-PN controlled grammar is unambiguous, then the algorithm is effective, that is, the time complexity is $O(n^3)$ where n is the length of a word. For ambiguous grammars, the algorithm is less effective, that is, the time complexity becomes $O(n^{k+4})$ for k-PN controlled grammars.

There are some restrictions on the algorithms investigated in this paper. The condition of no chain rules and no λ-rules is necessary to avoid infinitely many possibilities in derivation. Let us consider the next 1-PN controlled grammar shown in Fig. 3.

In the underlying grammar, there are infinite derivation trees for every word of the form $a(bb)^+$. For a fixed word in $a(bb)^+$, only one derivation is possible

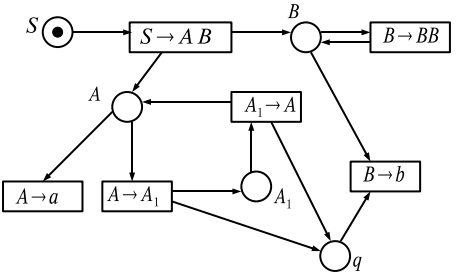

Fig. 3. A 1-PN controlled grammar with chain rules

under the control of the 1-Petri net. A parsing algorithm must select one possible derivation (tree) from infinite candidates. There may be two methods to resolve this problem: converting every k-PN controlled grammar into a grammar without chain rules and λ-rules or developing other algorithms for k-PN controlled grammars with chain rules or λ-rules. These will be done in future.

References

1. ter Beek, M., Kleijn, J.: Petri Net Control for Grammar Systems. In: Brauer, W., Ehrig, H., Karhumäki, J., Salomaa, A. (eds.) Formal and Natural Computing. LNCS, vol. 2300, pp. 220–243. Springer, Heidelberg (2002)
2. Dassow, J., Păun, G.: Regulated rewriting in formal language theory. Springer, Berlin (1989)
3. Dassow, J., Turaev, S.: k-Petri Net Controlled Grammars. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008. LNCS, vol. 5196, pp. 209–220. Springer, Heidelberg (2008)
4. Dassow, J., Turaev, S.: Petri net controlled grammars: the case of special Petri nets. Journal of Universal Computer Science 14, 2808–2835 (2009)
5. Dassow, J., Turaev, S.: Petri net controlled grammars: the power of labeling and final markings. Romanian Journal of Information Science and Technology 12, 191–207 (2009)
6. Dassow, J., Turaev, S.: Petri net controlled grammars with a bounded number of additional places. Acta Cybernetica 19, 609–634 (2010)
7. David, R., Alla, H.: Petri nets and grafcet: tool for modelling discrete event systems. Prentice Hall, Hertfordshire (1992)
8. Earley, J.: An efficient context-free parsing algorithm. Communications of the ACM 13, 94–102 (1970)
9. Hopcroft, J.H., Ullman, J.: Introduction to automata theory, languages, and computation. Addison-Wesley, Reading (1979)
10. Kasami, T.: An efficient recognition and syntax algorithm for context-free languages, Scientific Report AFCRL-65-758, Air force Cambridge Research Lab., Bedford, Mass (1965)

11. Reisig, W., Rozenberg, G. (eds.): APN 1998. LNCS, vol. 1491. Springer, Heidelberg (1998)
12. Rozenberg, G., Salomaa, A.: Handbook of Formal Languages, vol. 1-3. Springer, Berlin (1997)
13. Turaev, S.: Petri net controlled grammars. In: Proc. 3rd Doctoral Workshop on MEMICS 2007, Znojmo, Czech Republic, pp. 233–240 (2007)
14. Younger, D.H.: Recognition and parsing of context-free languages in time n^3. Information and Control 10, 189–208 (1967)