数理解析研究所講究録
第1799巻 2012年 49-56

Variants of Petri net controlled grammars and their parsing algorithms

Taishin Yasunobu Nishida[1]
Department of Information Sciences
Toyama Prefectural University

Abstract

We show an outline of three parsing algorithms for three variants of Petri net controlled grammars. Time complexities of these algorithm vary from deterministic polynomial time to nondeterministic polynomial time. These algorithms only parse grammars without λ-rules and cyclic rules.

1 Introduction and preliminaries

Recently Petri net controlled grammars have been introduced by M. ter Beek and J. Kleijn [1]. Then J. Dassow and S. Turaev have defined a number of variants of Petri net controlled grammars and they have investigated properties of languages generated by these variants [3, 4, 5, 6, 12]. But parsing algorithms for Petri net controlled grammars have not been considered.

In this extended abstract, we introduce some variants of Petri net controlled grammars and parsing algorithms for them. The simplest class, yet generates non-context-free languages, has a parsing algorithm of deterministic polynomial time. We introduce two other classes: one class has a parsing algorithm whose computation time is proportional to multiple of the number of derivation trees and their size and the other class has a parsing algorithm of nondeterministic polynomial time.

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, 10, 11].

1.1 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 $w_0 \overset{r_1 r_2 \cdots r_n}{\Longrightarrow} w_n$. A rule of the form $A \to \lambda$ where λ is the empty word is called a *λ-rule*. A set of rules $\{A_1 \to \alpha_1, \ldots, A_n \to \alpha_n\}$ is said to be *cyclic rules* if $\alpha_{i-1} = A_i$ for every $2 \leq i \leq n$ and $\alpha_n = A_1$. The language generated by G is defined by $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^*_G w\}$.

1.2 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

[1]email: nishida@pu-toyama.ac.jp

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 $\mu : P \to \mathbf{N}$, 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 $N = (P, T, F, \phi)$ 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 Petri net controlled grammars

In this section we define Petri net controlled grammars.

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 Petri net controlled grammar *is a quintuple $G = (V, \Sigma, S, R, N)$ where V, Σ, S, R are the components from the grammar G_0 and $N = (P', T', F', \phi', \iota')$ is a 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 which satisfy, 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.

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

Definition 2 *Let* $G = (V, \Sigma, S, R, N)$ *be a Petri net 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 *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 *enabled at the initial marking* ι *and finished at the marking* τ. *The* language generated *by* G, *denoted by* $L(G)$, *consists of all words which are successfully generated in* G.

Adding conditions on the set E of new arcs, we define the next variants of Petri net controlled grammars.

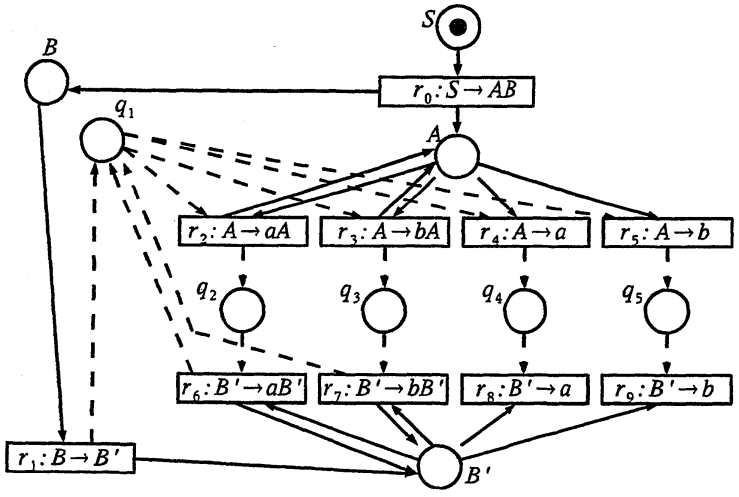

Figure 1: A tcPN controlled grammar generating $\{ww \mid w \in \{a,b\}^+\}$.

If the set of new arcs satisfies

- for every $t \in T$ if $(q,t) \in E$ and $(q',t) \in E$ for some $q, q' \in Q$, then $q = q'$ and
- for every $t \in T$ if $(t,q) \in E$ and $(t,q') \in E$ for some $q, q' \in Q$, then $q = q'$,

then the grammar G is said to be a *token conserving* Petri net controlled grammar, abbreviated to a tcPN controlled grammar. The condition restricts for every transition to have at most one arc to a place in Q and at most one arc from a place in Q.

Example 1 *Let* $G_1 = (\{S, A, B, B'\}, \{a, b\}, S, R, N)$ *be a tcPN controlled grammar where* R *and* N *is illustrated in Figure 1 in which rules are drawn in the rectangles of the corresponding transitions and arcs between transitions and control places are drawn in dashed lines. The grammar* G_1 *generates the language* $\{ww \mid w \in \{a, b\}^+\}$. □

If the set of new arcs of a tcPN controlled grammar satisfies that, for every $1 \leq i < j \leq k$, there exists no $t \in T$ such that $(t, q_i) \in E$ and $(q_j, t) \in E$, then the grammar G is said to be a *counter restricted* Petri net controlled grammar, or a crPN controlled grammar for short. The condition says that there are no cycles composed of places from Q, that is, places from Q only count number of applications of specific rules. The definition of crPN controlled grammars is identical to that of k-Petri net controlled grammars in [3, 6].

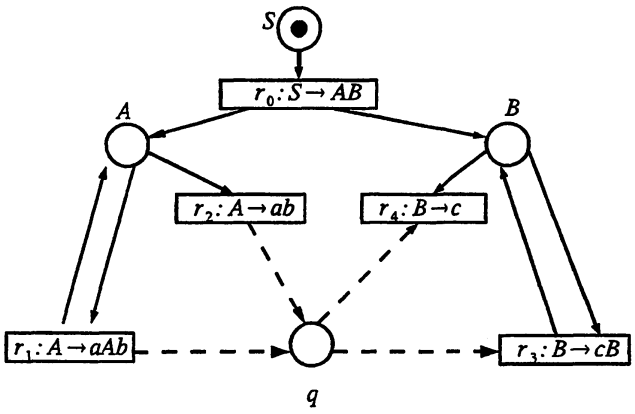

Figure 2: A scfPN controlled grammar generating $\{a^n b^n c^n \mid n > 0\}$.

If a crPN controlled grammar satisfies "every cycle in N does not contain any places in Q", then the grammar is called a *strict cycle free* Petri net controlled grammar, or an scfPN controlled grammar for short. A control Petri net of a crPN controlled grammar may have a cycle $(t_1, p_1), (p_1, t_2), \ldots, (p_n, t_1)$ in which $t_i \in T$ for every $1 \leq i \leq n$, $p_i \in P' (= P \cup Q)$ for every $1 \leq i \leq n$, and there exists some $p_j \in P$. A cycle in a control Petri net of an scfPN controlled grammar must satisfy that no node in any cycle is contained in Q.

Example 2 (Example 7 of [3]) *Let* $G_2 = (\{S, A, B\}, \{a, b, c\}, S, R, N)$ *be a scfPN controlled grammar where* N *is illustrated in Figure 2, The grammar* G_2 *generates the language* $\{a^n b^n c^n \mid n > 0\}$. □

A Petri net controlled grammar without any condition on the control Petri net N is called a *bijective* Petri net controlled grammar, abbreviated to a bPN controlled grammar, because there is a bijection from the set of transitions to the set of rules.

Example 3 *Let* $G_3 = (\{S', S, A, B, D, E, F\}, \{a, b, c\}, S', R, N)$ *be a bPN controlled grammar where* R *and* N *are illustrated in Figure 3. The grammar* G_3 *generates the language* $\{a^n b^n c^n\}^+$. □

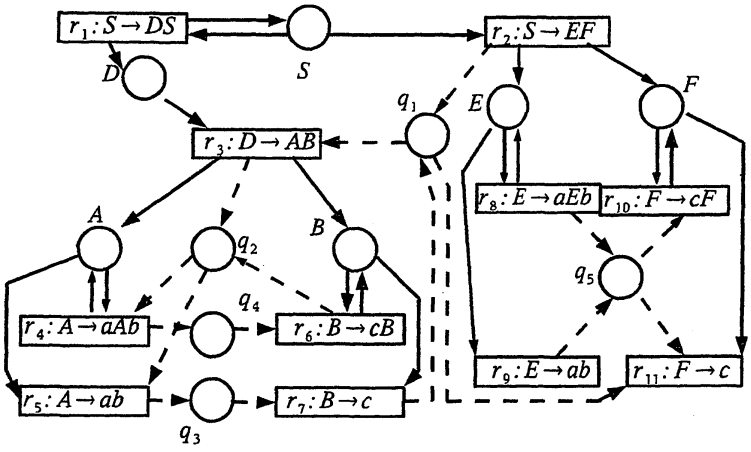

Figure 3: A bPN controlled grammar generating $\{a^n b^n c^n\}^+$.

3 Parsing algorithms

Now we show an outline of three algorithms which parse languages generated by scfPN controlled grammars, tcPN controlled grammars, and bPN controlled grammars. Before going to the algorithms, it should be noted that the algorithms shown here can parse Petri net controlled grammars with no λ-rules and no cyclic rules only.

3.1 scfPN controlled grammars

A parsing algorithm for scfPN controlled grammars is obtained by adding *token counters* to the Earley's algorithm for (usual) context-free grammars [8]. A token counter is a vector from $\mathbb{Z}^k$ where k is the number of control places (the places in Q) and $\mathbb{Z}$ is the set of all integers. The Earley's algorithm makes a state of the form $[A \to \alpha \cdot \beta]$ for a rule $A \to \alpha\beta$. The parsing algorithm for scfPN controlled grammars makes states with token counters by

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

The termination condition of the Earley's algorithm, the state $[S' \to S\cdot]$ is made as the state for the input word w where S' is a new start symbol with only rule $S' \to S$, is modified to that the state $[S' \to S\cdot]\vec{0}$ is made as the state for the input word where $\vec{0}$ is the zero vector in $\mathbb{Z}^k$.

Example 4 *In Example 2, the sets of Earley's states with token counters for the word $w = a^2b^2c^2$ are shown in the next table.*

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

□

3.2 tcPN controlled grammars

A parsing algorithm for tcPN controlled grammars makes a *derivation net* from the derivation tree which is made by a parsing algorithm (e.g., Earley's algorithm) for the underlying grammars. First the nodes in a derivation tree which are labelled by terminals are removed. The remaining nodes labelled by nonterminals and are replaced with rules which rewrite the nonterminals. Finally nodes of control places and arcs between control places and rules are added to the tree. A word generated by the underlying grammar is generated under the control of the token conserving Petri net if and only if all arcs between control places and rules in the derivation net form a collection of Eulerian paths.

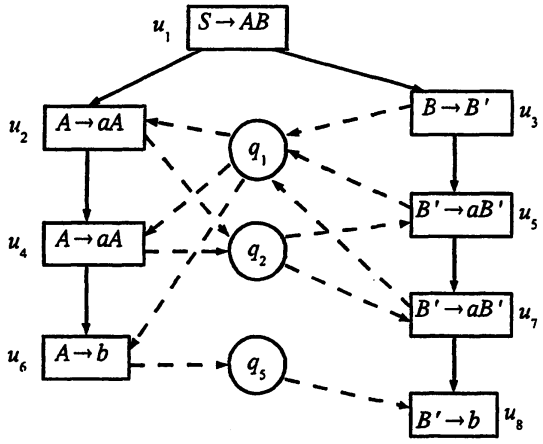

Figure 4: An example of a derivation net.

Example 5 *In Example 1, the derivation net for the word aabaab is shown in Figure 4. The derivation net has a collection of Eulerian paths.* □

3.3 bPN controlled grammars

A parsing algorithm for bPN controlled grammars makes a *conditional tree* for a input word. A conditional tree is based on the derivation tree for the underlying grammar. A conditional tree contains information of control places in every node as the form of $\bar{q}_i$ if the node is rewritten by a

rule which has an arc from a place q_i and q_j if the node is rewritten by a rule which has an arc to a place q_j. Next $\bar{q}_i$ and q_i is cancelled if they satisfy a condition which is equivalent to the movement of tokens in the control Petri net. A word is generated by a bPN controlled grammar if and only if the conditional tree for the word is cancelled to an empty tree.

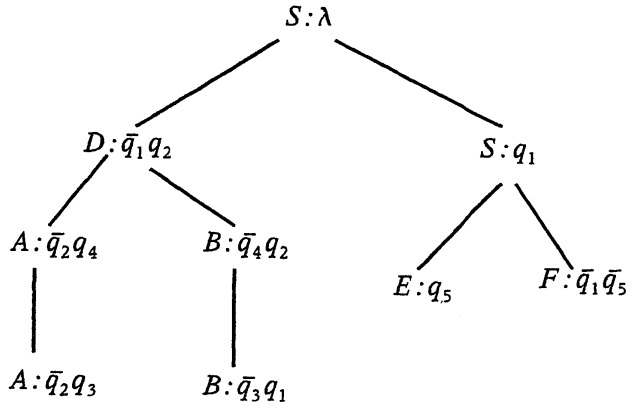

Figure 5: An example of a conditional tree.

Example 6 *In Example 3, the conditional tree for the word $a^2b^2c^2abc$ is shown in Figure 5. The conditional tree can be cancelled to an empty tree.* □

4 Conclusion

We have introduced parsing algorithms for PN controlled grammars. If the underlying grammar is unambiguous, that is, there is just one derivation tree for every word, then the algorithms for scfPN controlled grammars and tcPN controlled grammars are effective. For input word of length n, the time complexities of the algorithms for scfPN controlled grammars and tcPN controlled grammars are $O(n^3)$. For ambiguous grammars, the algorithms for scfPN controlled grammars and tcPN controlled grammars are less effective, $O(n^{k+4})$ for scfPN controlled grammars where k is the number of places in Q and $O(2^n)$ for tcPN controlled grammars.

On the other hand, the algorithm for bPN controlled grammars is nondeterministic and computes in polynomial time of the length of the input. Of course a deterministic version with exponential time is easily constructed.

Let $\mathcal{L}$(xPN) be a class of languages generated by xPN controlled grammars where x $\in$ {b, tc, cr, scf}. By definition we have

$$\mathcal{L}(\text{scfPN}) \subseteq \mathcal{L}(\text{crPN}) \subseteq \mathcal{L}(\text{tcPN}) \subseteq \mathcal{L}(\text{bPN}).$$

It does not known whether some of above inclusions are proper or not. In [4], the most general Petri net controlled grammars, with no restriction on a Petri net, or equivalently, the mapping from the set of transitions in the control Petri net to the set of rules is not necessarily bijective, are defined and it has been shown that the class of languages generated by such grammars is equivalent to the class of languages generated by matrix grammars. Therefore, the class of languages generated by

bPN controlled grammars is included in the class of matrix grammars. Precise characterization of the classes shown above is a future work.

References

[1] Maurice ter Beek and Jetty Kleijn, Petri net control for grammar systems, in W. Brauer, H. Ehrig, J. Karhumäki, and A. Salomaa (eds.) Formal and Natural Computing, LNCS 2300, 220-243, (Springer, Berlin, 2002).

[2] Jürgen Dassow and Gheorghe Păun, *Regulated rewriting in formal language theory*, (Springer, Berlin 1989).

[3] Jürgen Dassow and Sherzod Turaev, k-Petri net controlled grammars, in C. Martín-Vide, F. Otto, and H. Fernau (eds.) LATA 2008, LNCS 5196, 209–220 (Springer, Berlin, 2008).

[4] Jürgen Dassow and Sherzod Turaev, Petri net controlled grammars: the case of special Petri nets, *Journal of Universal Computer Science*, 2808–2835, Vol. **14** (2009).

[5] Jürgen Dassow and Sherzod Turaev, Petri net controlled grammars: the power of labeling and final markings, *Romanian Journal of Information Science and Technology*, 191–207, Vol.**12** (2009).

[6] Jürgen Dassow and Sherzod Turaev, Petri net controlled grammars with a bounded number of additional places, *Acta Cybernetica*, 609–634, Vol. **19** (2010).

[7] René David and Hassan Alla, *Petri nets and grafcet: tool for modelling discrete event systems*, (Prentice Hall, Hertfordshire, 1992).

[8] Jay Earley, An efficient context-free parsing algorithm, *Communications of the ACM*, 94–102, Vol. **13** (1970).

[9] John H. Hopcroft and Jeffrey Ullman, *Introduction to automata theory, languages, and computation*, (Addison-Wesley, Reading, 1979).

[10] Wolfgang Reisig and Grzegorz Rozenberg (eds), *Lectures on Petri nets I: Basic model*, LNCS 1491, (Springer, Berlin, 1998).

[11] Grzegorz Rozenberg, Arto Salomaa, *Handbook of Formal Languages* vol. 1 – 3, Springer, Berlin, 1997.

[12] Sherzod Turaev, Petri net controlled grammars, *Proc. 3rd Doctoral Workshop on MEMICS-2007*, 233–240, (Znojmo, Czech Republic, 2007).