Gairatzhan Mavlankulov; Mohamed Othman; Sherzod Turaev; Mohd Hasan Selamat; Laula Zhumabayeva; Tamara Zhukabayeva

Concurrently controlled grammars

Kybernetika, Vol. 54 (2018), No. 4, 748–764

Persistent URL: http://dml.cz/dmlcz/147422

KYBERNETIKA — VOLUME *54* (2018), NUMBER 4, PAGES 748–764

CONCURRENTLY CONTROLLED GRAMMARS

Gairatzhan Mavlankulov, Mohamed Othman, Sherzod Turaev,
Mohd Hasan Selamat, Laula Zhumabayeva, and Tamara Zhukabayeva

This paper introduces a new variant of Petri net controlled grammars, namely a *concurrently controlled grammar*, where the control over the application of the productions of a grammar is realized by a Petri net with different parallel firing strategies. The generative capacity of these grammars is investigated with respect to transition labeling strategies, definitions of final marking sets and parallel transition firing modes. It is shown that the labeling strategies do not effect the computational power whereas the maximal firing modes increase the power of concurrently controlled grammars with erasing rules up to Turing machines.

Keywords: parallel computing, controlled grammars, Petri net, concurrent grammars

Classification: 68Q45, 68Q85

1. INTRODUCTION

Petri nets [26], "dynamic" bipartite directed graphs with two sets of nodes, called *places* and *transitions*, provide an elegant and powerful mathematical formalism for modelling concurrent systems and their behaviour. Since Petri nets successfully describe and analyse the flow of information and the control of action in such systems, they can be very suitable tools for studying the properties of formal languages. Initially, used as language generating/accepting tools [14, 15, 17, 18, 20, 35, 36], eventually, Petri Nets have been widely applied as regulation mechanisms for grammar systems [1], automata [2, 11, 12, 13, 19, 37, 38], and grammars [3, 4, 5, 6, 7, 8, 9, 10, 16, 22, 27, 28, 29, 30, 31, 32, 33, 34]. The concept of maximal parallelism in Petri nets is studied in [2]. In [11, 13] authors investigate another different viewpoints on the parallel firing of transitions in Petri nets. They introduce Turing Machines with Petri nets as finite control (called Concurrent Turing Machines), where each token within a marking is associated with an individual read and write head on a tape of Concurrent Turing Machine. In [13] researchers define and study the variant of the Concurrent Finite Automaton (CFA), where finite automata are generalized by using Petri nets as control. Some modes of firing transitions in Petri nets are compared and investigated in [21]. A generalization of regularly controlled grammars (called as *Petri net controlled grammar*) is considered in [4, 9]: instead of a finite automaton a Petri net is associated with a context-free grammar and it is required that the sequence of applied rules corresponds to an occurrence sequence of the Petri

DOI: 10.14736/kyb-2018-4-0748

net, i. e., to sequences of transitions which can be fired in succession. Several variants of Petri net controlled grammars have been introduced and investigated: In [5, 6, 10] authors introduce k-Petri net controlled grammars and study their properties including generative power, closure properties, infinite hierarchies. In [7, 8] researchers investigate grammars controlled by the structural subclasses of Petri nets, namely state machines, marked graphs, causal nets, free-choice nets, asymmetric choice nets and ordinary nets. They prove that the family of languages generated by (arbitrary) Petri net controlled grammars coincide with the family of languages generated by grammars controlled by free-choice nets. The continuation of the research on Petri net controlled grammars by restricting to (context-free, extended or arbitrary) Petri nets with place capacities have been investigated in [28, 29, 30]. A Petri net with place capacity regulates the defining grammar by permitting only those derivations where the number of each nonterminal in each sentential form is bounded by its capacity. It was shown that several families of languages generated by grammars controlled by extended cf Petri nets with place capacities coincide with the family of matrix languages of finite index. In [23, 24, 25] authors introduce and study the conception of context free concurrent grammars, where grammars are controlled by context free Petri nets under parallel firing strategies, i. e., the transitions of a Petri net fire simultaneously in different modes.

In this paper, we introduce new variants of Petri net controlled grammars, called *concurrently controlled grammars*, where we use p/t Petri nets under different parallel firing strategies, and study their generative powers. We consider concurrently controlled grammars with various firing modes (multistep, maximal multistep, labeled multistep), labeling strategies (free labeling, arbitrary labeling, extended labeling) and definitions of final marking sets (t-type, r-type), which result in different classes of languages generated by such grammars. We show that the computational power of concurrently controlled grammars is at least as powerful as their sequential counterparts, on the other hand, the generative power can be increased up to the power of Turing Machines under a specific firing mode and labeling strategy. This work is a new approach in studying theoretical models for concurrent and parallel computation.

2. PRELIMINARIES

Let $\mathbb{N}$ be the set of all non-negative integers and $\mathbb{N}^k$ be the set of all vectors of k non-negative integers. The cardinality of a set X is denoted by $|X|$.

2.1. Grammars and languages

Let Σ be an *alphabet* which is a finite nonempty set of symbols. A *string* over the alphabet Σ is a finite sequence of symbols from Σ. The *empty* string is denoted by λ. The set of all strings over the alphabet Σ is denoted by Σ^*. A subset of Σ^* is called a *language*. The *length* of a string w, denoted by $|w|$, is the number of occurrences of symbols in w. The number of occurrences of a symbol a in a string w is denoted by $|w|_a$.

A *context-free grammar* is a quadruple $G = (V, \Sigma, S, R)$ where V and Σ are the disjoint finite sets of *nonterminal* and *terminal* symbols, respectively, $S \in V$ is the *start* symbol and $R \subseteq V \times (V \cup \Sigma)^*$ is a finite set of *(production) rules*. Usually, a rule (A, x) is written as $A \to x$. A and x are respectively called the *left* and *right sides* of the

production. A rule of the form $A \to \lambda$ is called an *erasing rule*. A grammar G is called λ-*free* if all the productions of R are non-erasing, or $S \to \lambda$ is the only erasing rule and S does not appear on the right side of any production.

For $x \in (V \cup \Sigma)^*$ and $y \in (V \cup \Sigma)^*$, we write $x \Rightarrow y$, iff there is a rule $r = A \to \beta \in R$ such that $x = x_1 A x_2$ and $y = x_1 \beta x_2$. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. A derivation using the sequence of rules $\pi = r_1 r_2 \cdots r_n$ is denoted by $\overset{\pi}{\Longrightarrow}$ or $\overset{r_1 r_2 \cdots r_n}{\Longrightarrow}$.

The *language* generated by G is defined by $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$. The family of context-free languages is denoted by **CF**.

2.2. Multisets

A *multiset* over an alphabet Σ is a mapping $\mu : \Sigma \to \mathbb{N}$. The set Σ is called the *basic set* of a multiset μ and the elements of Σ is called the *basic elements* of a multiset μ. A multiset μ over an alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$ can be denoted by $\mu = (\mu(a_1)a_1, \mu(a_2)a_2, \ldots, \mu(a_n)a_n)$ where $\mu(a_i)$, $1 \le i \le n$, is the multiplicity of a_i, or as a vector $\mu = (\mu(a_1), \mu(a_2), \ldots, \mu(a_n))$,

The empty multiset is denoted by ϕ, that is $\phi(a) = 0$ for all $a \in \Sigma$. The set of all multisets over Σ is denoted by $\Sigma^{\oplus}$.

For two multisets $\mu_1, \mu_2 \in \Sigma^{\oplus}$, we define

— the *inclusion* by $\mu_1 \sqsubseteq \mu_2$ if and only if $\mu_1(a) \le \mu_2(a)$ for all $a \in \Sigma$;

— the *sum* by $(\mu_1 \oplus \mu_2)(a) = \mu_1(a) + \mu_2(a)$ for each $a \in \Sigma$;

— the *difference* by $(\mu_1 \ominus \mu_2)(a) = \max\{0, \mu_1(a) - \mu_2(a)\}$ for each $a \in \Sigma$.

2.3. Petri nets

A *Petri net* (*PN* for short) is a quadruple (P, T, ξ_1, ξ_2) where P and T are finite disjoint sets of *places* and *transitions*, respectively, mappings $\xi_1, \xi_2 : T^{\oplus} \to P^{\oplus}$ are homomorphisms where $\xi_1(t)$ and $\xi_2(t)$ define pre- and post- multisets of transition $t \in T$, respectively. A multiset $\mu \in P^{\oplus}$ is called a *marking*.

A *labeled Petri net* is a tuple $K = (\Sigma, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ where Σ is an alphabet, (P, T, ξ_1, ξ_2) is a Petri net, $\gamma : T \to \Sigma \cup \{\lambda\}$ is a transition labeling function, $\mu_0 \in P^{\oplus}$ is the *initial marking*, and $M \subseteq P^{\oplus}$ is a finite set of final markings.

A transition $t \in T$ is called λ-*transition* if and only if $\gamma(t) = \lambda$. A transition labeling function γ is called λ-*free* if no transition is labeled with the empty string, i. e., $\gamma(t) \ne \lambda$ for all $t \in T$, and *arbitrary* if no restriction is posed on it.

An *occurrence relation* $\overset{t}{\to}$ over $\Sigma^* \times P^{\oplus}$ is defined by $(x, \mu_1) \overset{t}{\to} (y, \mu_2)$ if and only if

$$y = x\gamma(t),\ \xi_1(t) \sqsubseteq \mu_1 \text{ and } \mu_2 = (\mu_1 \ominus \xi_1(t)) \oplus \xi_2(t)$$

where $x, y \in \Sigma^*$ and $\mu_1, \mu_2 \in P^{\oplus}$.

An *occurrence sequence* from (x_0, μ_0) to (x_k, μ_k) is a finite sequence of transitions $t_1 t_2 \cdots t_k \in T^*$, $t_i \in T$, $1 \le i \le k$, such that

$$(x_0, \mu_0) \xrightarrow{t_1} (x_1, \mu_1) \xrightarrow{t_2} \cdots \xrightarrow{t_k} (x_k, \mu_k).$$

In short this sequence can be written as $(x_0, \mu_0) \xrightarrow{t_1 t_2 \cdots t_k} (x_k, \mu_k)$ or $(x_0, \mu_0) \xrightarrow{\tau} (x_k, \mu_k)$ where $\tau = t_1 t_2 \cdots t_k$.

The *language* of a labeled Petri net K with respect to a transition labeling function γ and a final marking set M is defined by

$$L(K) = \{w \in \Sigma^* \mid (\lambda, \mu_0) \xrightarrow{\tau} (w, \mu) \text{ where } \tau \in T^* \text{ and } \mu \in M\}.$$

2.4. Petri net controlled grammars

Definition 2.1. A *Petri net controlled grammar* (*Petri net grammar*, for short) is a tuple $\mathcal{G} = (V, \Sigma, S, R, K)$ where (V, Σ, S, R) is a context-free grammar, and $K = (R, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ is a labeled Petri net.

Definition 2.2. A *Petri net controlled grammar* $\mathcal{G}$ is called (1) λ-*free* (denoted by $-\lambda$) if γ is λ-free, (2) *arbitrary* (denoted by λ) if γ is arbitrary, (3) r-*type* (denoted by r) if $M = P^{\oplus}$, and (4) t-*type* (denoted by t) if M is a finite set.

We use the notation (x, y)-*Petri net grammar* where $x \in \{-\lambda, \lambda\}$ shows the type of a labeling function and $y \in \{r, t\}$ shows the type of a set of final markings.

Definition 2.3. A *derivation step* is a binary relation $\overset{t}{\Longrightarrow}$ over $(V \cup \Sigma)^* \times P^{\oplus}$ defined by $(w_1, \mu_1) \overset{t}{\Longrightarrow} (w_2, \mu_2)$ if and only if $\xi_1(t) \sqsubseteq \mu_1$, $\mu_2 = (\mu_1 \ominus \xi_1(t)) \oplus \xi_2(t)$ and one of the following conditions holds

(1) $\gamma(t) = A \to \beta \in R$ and there are strings $x_1, x_2 \in (V \cup \Sigma)^*$ such that $w_1 = x_1 A x_2$ and $w_2 = x_1 \beta x_2$,

(2) $\gamma(t) = \lambda$ and $w_1 = w_2$.

Definition 2.4. A *derivation* from (w, μ) to (w', μ') is a sequence of derivation steps

$$(w_0, \mu_0) \overset{t_1}{\Longrightarrow} (w_1, \mu_1) \overset{t_2}{\Longrightarrow} \cdots \overset{t_k}{\Longrightarrow} (w_k, \mu_k)$$

where $w_0 = w$, $w_k = w'$, $\mu_0 = \mu$, $\mu_k = \mu'$, and $t_1 t_2 \cdots t_k \in T^*$.

For short this sequence can be written as $(w, \mu) \overset{t_1 t_2 \cdots t_k}{\Longrightarrow} (w', \mu')$ or $(w, \mu) \overset{\tau}{\Longrightarrow} (w', \mu')$ where $\tau = t_1 t_2 \cdots t_k$.

Definition 2.5. The *language* generated by a Petri net controlled grammar $\mathcal{G}$ is defined by

$$L(\mathcal{G}) = \{w \in \Sigma^* \mid (S, \mu_0) \overset{\tau}{\Longrightarrow} (w, \mu) \text{ where } \tau \in T^* \text{ and } \mu \in M\}.$$

We denote by $\mathbf{PN}(x, y)$ and $\mathbf{PN}^{\lambda}(x, y)$ the families of languages generated by (x, y)-PN grammars without and with erasing rules, respectively, where $x \in \{-\lambda, \lambda\}$ and $y \in \{r, t\}$.

The relations of these families of languages with respect to $\mathbf{MAT}$ and $\mathbf{MAT}^{\lambda}$, the families of matrix languages, is depicted in the following theorem.

Theorem 2.6. (Zetzsche [37]) For $x \in \{-\lambda, \lambda\}$,

$$\mathbf{MAT} \subseteq \mathbf{PN}(x, r) \subseteq \mathbf{PN}(x, t) = \mathbf{PN}^{\lambda}(x, t) = \mathbf{MAT}^{\lambda}.$$

2.5. Multisteps

We recall some notions and notations related to "multistep" in Petri nets.

Let $K = (\Sigma, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ be a labeled Petri net, and let $T_x = \{t \mid \gamma(t) = x\}$, for some $x \in \Sigma$. A multiset $\tau \in (T_x)^{\oplus}$ is called a *labeled multistep* (simply *multistep*).

For strings $w_1, w_2 \in \Sigma^*$, markings $\mu_1, \mu_2 \in P^{\oplus}$ and a multistep $\tau \in (T_x)^{\oplus}$, a *multistep occurrence* is defined by $(w_1, \mu_1) \xrightarrow{\tau}_m (w_2, \mu_2)$ if and only if

$$w_2 = w_1 x \text{ and } \xi_1(\tau) \sqsubseteq \mu_1, \mu_2 = (\mu_1 \ominus \xi_1(\tau)) \oplus \xi_2(\tau).$$

A multistep τ is called

- *maximal in* μ_1 if and only if, for all $\tau' \in (T_x)^{\oplus}$, $\tau \sqsubseteq \tau'$ and $\xi_1(\tau') \sqsubseteq \mu_1$ imply $\tau' = \tau$,
- a *step* if τ is a multiset, where its each element has a multiplicity of one, i. e., $\tau \subseteq T_x$,
- a *maximal step in* μ_1 if and only if, for all $\tau' \subseteq T_x$, $\tau \sqsubseteq \tau'$ and $\xi_1(\tau') \sqsubseteq \mu_1$ imply $\tau' = \tau$.

A step (maximal multistep, maximal step) occurrence is denoted by $\xrightarrow{\tau}_s$ ($\xrightarrow{\tau}_{\widehat{m}}$ and $\xrightarrow{\tau}_{\widehat{s}}$, respectively). The reflexive transitive closure of $\to_\rho$ is denoted by $\xrightarrow{*}_\rho$.

The *languages* of a labeled Petri net K with steps, multisteps, maximal steps and maximal multisteps are defined by

$$L_\rho(K) = \{w \in \Sigma^* \mid (\lambda, \mu_0) \xrightarrow{*}_\rho (w, \mu) \text{ where } \mu \in M\}, \rho \in \{s, m, \widehat{s}, \widehat{m}\}.$$

A language $L_\rho(K)$ is called λ-free if $\gamma(t) \neq \lambda$ for all $t \in T$. The family of (λ-free) languages of type $\rho \in \{s, m, \widehat{s}, \widehat{m}\}$ is denoted by $\mathcal{L}_\rho^\lambda$ ($\mathcal{L}_\rho$).

The main relations of these languages families are demonstrated in the following theorem (for details, see [21]):

Theorem 2.7.

(1) $\mathcal{L}_s \subset \mathcal{L}_m \subset \mathcal{L}_{\widehat{m}} \subset \mathbf{CS} \subset \mathbf{RE} = \mathcal{L}_{\widehat{s}}^\lambda = \mathcal{L}_{\widehat{m}}^\lambda$,

(2) $\mathcal{L}_s \subset \mathcal{L}_{\widehat{s}} \subset \mathcal{L}_{\widehat{m}}$,

(3) $\mathcal{L}_m \subseteq \mathcal{L}_s^\lambda = \mathcal{L}_m^\lambda \subset \mathbf{RE}$,

where **CS** and **RE** are respectively the families of context-sensitive and recursively enumerable languages.

3. CONCURRENT GRAMMARS

Definition 3.1. Let $\mathcal{G} = (V, \Sigma, S, R, K)$ be a Petri net grammar where $K = (R, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$. For $r = A \to \beta \in R$, let $T_r = \{t \in T \mid \gamma(t) = r\}$, and let $T_\lambda = \{t \in T \mid \gamma(t) = \lambda\}$. We say that $(w_1, \mu_1) \in (V \cup \Sigma)^* \times P^\oplus$ *directly derives* $(w_2, \mu_2) \in (V \cup \Sigma)^* \times P^\oplus$ *with a multistep* τ, written as $(w_1, \mu_1) \overset{\tau}{\Longrightarrow}_\rho (w_2, \mu_2)$, if and only if $\xi_1(\tau) \sqsubseteq \mu_1$, $\mu_2 = (\mu_1 \ominus \xi_1(\tau)) \oplus \xi_2(\tau)$ and one of the following conditions holds

(1) $\tau \in (T_r)^\oplus$ and there are strings $x_1, x_2 \in (V \cup \Sigma)^*$ such that $w_1 = x_1 A x_2$ and $w_2 = x_1 \beta x_2$,

(2) $\tau \in (T_\lambda)^\oplus$ and $w_1 = w_2$.

Then a relation $\overset{\tau}{\Longrightarrow}_\rho$ is called a *derivation step in ρ-mode* where $\rho \in \{s, m, \widehat{s}, \widehat{m}\}$.

Definition 3.2. A Petri net grammar $\mathcal{G}$ with derivations in any ρ-mode, $\rho \in \{s, m, \widehat{s}, \widehat{m}\}$, is called a *concurrent grammar*.

We use a clearer notation (x, y, ρ)-*concurrent grammar* where $x \in \{-\lambda, \lambda\}$ shows the type of a labeling function, $y \in \{r, t\}$ the type of a set of final markings, and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$ the derivation mode.

Definition 3.3. The *language* generated by a concurrent grammar $\mathcal{G}$ in ρ-mode, $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$, is defined as

$$L_\rho(\mathcal{G}) = \{w \in \Sigma^* \mid (S, \mu_0) \overset{\tau_1 \tau_2 \cdots \tau_k}{\Longrightarrow}_\rho (w, \mu), \text{ where } \tau_i \in (T_{x_i})^\oplus, \\ x_i \in R \cup \{\lambda\}, 1 \le i \le k, \text{ and } \mu \in M\}.$$

The language generated by a concurrent grammar is called a *concurrent language*.

We denote by $\mathbf{PN}(x, y, \rho)$ and $\mathbf{PN}^\lambda(x, y, \rho)$ the families of languages generated by (x, y, ρ)-concurrent grammars without and with erasing rules, respectively. We also use bracket notation $\mathbf{PN}^{[\lambda]}(x, y, \rho)$ in order to say that a statement holds both in case with erasing rules and in case without erasing rules.

Further, we define a normal form for concurrent grammars which is needed in sequel.

Definition 3.4. A concurrent grammar $\mathcal{G}$ is said to be in *nonterminal normal form* if all productions are of the form

- $A \to \beta$ where $\beta \in V^*$ or
- $A \to a$ where $a \in \Sigma$.

Lemma 3.5. For every concurrent grammar $\mathcal{G}$ there exists an equivalent concurrent grammar $\mathcal{G}'$ in nonterminal normal form.

P r o o f . Such a normal form can be easily constructed from arbitrary concurrent grammar by replacing each terminal a on the right side of each production with a nonterminal Z_a, and introducing the production $Z_a \to a$. The labeled Petri net is modified accordingly: the transitions of the labeled Petri net will preserve the corresponding modified productions, and we add new transitions and the connecting place for all productions $Z_a \to a$.

Let $\mathcal{G} = (V, \Sigma, S, R, K)$ be an (x, y, ρ)-concurrent grammar with $K = (R, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ where $x \in \{-\lambda, \lambda\}$, $y \in \{r, t\}$, and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$.

Let $Z_\Sigma = \{Z_a \mid a \in \Sigma\}$ be a set of new nonterminals. We define a coding $h : (V \cup \Sigma) \to (V \cup Z_\Sigma)$ as $h(A) = A$ for all $A \in V$ and $h(a) = Z_a$ for all $a \in \Sigma$. Then the grammar $\mathcal{G}' = (V', \Sigma, S', R', K')$ with $K' = (R', P', T', \xi_1', \xi_2', \gamma', \mu_0', M')$ is defined as follows:

- $V' = V \cup Z_\Sigma \cup \{S'\}$ where $S' \notin V \cup Z_\Sigma$,
- $R' = \{A \to h(\beta) \mid A \to \beta \in R\} \cup \{Z_a \to a \mid a \in \Sigma\} \cup \{S' \to S\}$,
- $P' = P \cup \{p_0, p_1\}$,
- $T' = T \cup \{t_0\} \cup \{t_a \mid a \in \Sigma\}$,
- $M' = P'^{\oplus}$, (if M is finite, $M' = \{\nu_\mu | \mu \in M\}$,
- for all $t \in T'$,
$$\xi_1'(t) = \begin{cases} \xi_1(t) & \text{if } t \in T, \\ \{p_0\} & \text{if } t = t_0, \\ \{p_1\} & \text{if } t = t_a \text{ for all } a \in \Sigma, \end{cases}$$
- for all $t \in T'$,
$$\xi_2'(t) = \begin{cases} \xi_2(t) & \text{if } t \in T, \\ P_0 & \text{if } t = t_0, \\ \{p_1\} & \text{if } t = t_a \text{ for all } a \in \Sigma, \end{cases}$$
where $P_0 = \mu_0 \oplus p_1$,
- for all $p \in P'$,
$$\mu_0'(p) = \begin{cases} 0 & \text{if } p \in P \cup \{p_1\}, \\ 1 & \text{if } p = p_0, \end{cases}$$
- for all $p \in P'$, and for each $\nu_\mu \in M'$ where $\mu \in M$,
$$\nu_\mu(p) = \begin{cases} \mu(p) & \text{if } p \in P, \\ 0 & \text{if } p = p_0, \\ 1 & \text{if } p = p_1. \end{cases}$$

Any derivation of a string w in $\mathcal{G}$ can be easily simulated by a derivation of $\mathcal{G}'$: the derivation starts with $S' \to S$, and a sentential form produced by $A \to \beta \in R$ is replaced with the sentential form produced by $A \to h(\beta)$. Then, $h(w)$ is resulted. Further, productions of the form $Z_a \to a$ are applied. By construction, since, at a time, only one t_a can be fired, the firing mode is not affected. Thus, $L(\mathcal{G}) \subseteq L(\mathcal{G}')$. The inverse inclusion is also can be shown by using similar arguments. □

4. RESULTS

In this section, we study the effects of the labeling strategy and the definitions of final marking sets to the generative power of concurrent grammars. Moreover, we discuss the effect of firing modes to the generative capacity of concurrent grammars, and establish the relations among different families of concurrent languages.

4.1. The power of labelings

In this subsection, we study the effects of the labeling strategy to the generative power of concurrent grammars. From the definition, the next lemma follows immediately.

Lemma 4.1. For all $y \in \{r, t\}$ and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$,

$$\mathbf{PN}^{[\lambda]}(-\lambda, y, \rho) \subseteq \mathbf{PN}^{[\lambda]}(\lambda, y, \rho).$$

Next, we show that the inverse inclusion also holds.

Lemma 4.2. For all $y \in \{r, t\}$ and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$,

$$\mathbf{PN}^{[\lambda]}(\lambda, y, \rho) \subseteq \mathbf{PN}^{[\lambda]}(-\lambda, y, \rho).$$

Proof. Let $\mathcal{G} = (V, \Sigma, S, R, K)$ be a (λ, y, ρ)-concurrent grammar, where $K = (R, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$. Following the proof of Lemma 3.5, first, we construct an equivalent $(-\lambda, y, \rho)$-concurrent grammar $\mathcal{G}' = (V', \Sigma, S', R', K')$ with $K' = (R', P', T', \xi_1', \xi_2', \gamma', \mu_0', M')$.

The idea of the definition of the equivalent $(-\lambda, y, \rho)$-concurrent grammar $\mathcal{G}'' = (V', \Sigma, S', R'', K'')$ where $K'' = (R'', P', T'', \xi_1'', \xi_2'', \gamma'', \mu_0', M')$ has the same construction as K' except additional copies of λ-transitions with the incoming and outgoing edges. Let $T_\lambda = \{t \in T \mid \gamma(t) = \lambda\}$. For each λ-transition $t \in T_\lambda$, we introduce $|V \cup \Sigma|$ "copies" t_A for all $A \in V$, t_a for all $a \in \Sigma$, whose labels correspondingly $A \to A$ and $T_a \to T_a$. Each copy of $t \in T_\lambda$ has the same input and output places as t. The modified components of $\mathcal{G}''$ are defined as follows:

- $R'' = R' \cup \{A \to A \mid A \in V'\} \mid a \in \Sigma\}$,
- $T'' = T' \cup \{t_{\lambda,A} \mid t \in T_\lambda \text{ and } A \in V\} \cup \{t_{\lambda,a} \mid t \in T_\lambda \text{ and } a \in \Sigma\}$,
- for all $t \in T''$ and $i = 1, 2$,

$$\xi_i''(t) = \begin{cases} \xi_i'(t) & \text{if } t \in T' - T_\lambda, \\ \xi_i'(t_\lambda) & \text{if } t = t_{\lambda,A} \text{ and } t_\lambda \in T_\lambda, \\ \xi_i'(t_\lambda) & \text{if } t = t_{\lambda,a} \text{ and } t_\lambda \in T_\lambda, \end{cases}$$

- for all $t \in T''$,

$$\gamma''(t) = \begin{cases} \gamma(t) & \text{if } t \in T' - T_\lambda, \\ A \to A & \text{if } t = t_{\lambda,A} \text{ for all } A \in V, \\ T_a \to T_a & \text{if } t = t_{\lambda,a} \text{ for all } a \in \Sigma. \end{cases}$$

In every time when non-λ-transitions in K' are fired, the identical transitions are also fired in K'', which result in a sentential form containing some nonterminals A from $(V \cup T_\Sigma)$. If, further, some λ-transitions are fired in K', then, in K'', we choose those copies of these λ-transitions whose labels are $A \to A$. In derivations, where used the productions $S \to \lambda$, we cannot replace λ-transitions by transitions labeled $A \to A$ (or $T_a \to T_a$), because a production $S \to S$ is not allowed in a non-erasing grammar. In this case we remove the production $S \to \lambda$ from the original grammar, then apply the above construction. If the original grammar can generate λ, then constructed grammar can be easily modified by adding λ to the language. Thus, it is not difficult to see that $L(\mathcal{G}) = L(\mathcal{G}') = L(\mathcal{G}'')$. □

Remark 4.3. We notice that it is undecidable whether a language $\mathbf{PN}(\lambda, \widehat{m}, t)$ contains λ. Since the equality $\mathbf{RE} = \mathcal{L}^{\lambda}_{\widehat{m}}$ from [21] is effective, it is not decidable whether $a \in L$ for a given $L \in \mathcal{L}^{\lambda}_{\widehat{m}}$. Moreover, it is decidable weather a language in $\mathbf{PN}(-\lambda, \widehat{m}, t)$ contains λ. This amounts to deciding the membership problem for a language in $\mathcal{L}_{\widehat{m}}$, which is decidable. Thus, it is not possible to effectively translate a $PN(\lambda, \widehat{m}, t)$-grammar into an equivalent $PN(-\lambda, \widehat{m}, t)$ -grammar.

From the Lemma 4.1 and Lemma 4.2, we obtain the following result

Theorem 4.4. The labeling strategy does not effect to the generative capacity of concurrent grammars, i. e., for all $y \in \{r, t\}$ and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$,

$$\mathbf{PN}(-\lambda, y, \rho) = \mathbf{PN}(\lambda, y, \rho) \subseteq \mathbf{PN}^{\lambda}(-\lambda, y, \rho) = \mathbf{PN}^{\lambda}(\lambda, y, \rho).$$

4.2. The power of final markings

In this subsection, we study the effect of the definitions of final marking sets to the generative power.

Lemma 4.5. For all $x \in \{-\lambda, \lambda\}$ and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$,

$$\mathbf{PN}^{[\lambda]}(x, r, \rho) \subseteq \mathbf{PN}^{[\lambda]}(x, t, \rho).$$

Proof. Let $\mathcal{G} = (V, \Sigma, S, R, K)$ with $K = (R, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ be an (x, r, ρ)-concurrent grammar where $x \in \{-\lambda, \lambda\}$ and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$. Following the proof of Lemma 4.2, first, we construct an equivalent $(-\lambda, r, \rho)$-concurrent grammar $\mathcal{G}' = (V', \Sigma, S', R', K')$ with a labeled Petri net $K' = (R', P', T', \xi'_1, \xi'_2, \gamma', \mu'_0, M')$, which is in nonterminal normal form.

We define the $(-\lambda, t, \rho)$-concurrent grammar $\mathcal{G}'' = (V', \Sigma, S', R', K'')$ with $K'' = (R', P', T'', \xi''_1, \xi''_2, \gamma'', \mu'_0, M'')$ such that $L_\rho(\mathcal{G}) = L_\rho(\mathcal{G}') = L_\rho(\mathcal{G}'')$. For each $p \in P'$, we construct the set $T_{p,\Sigma'}$ and $T_{p,V'}$, where $T_{p,\Sigma'} = \{t_{p,a} \mid a \in \Sigma'\}$ and $T_{p,V'} = \{t_{p,A} \mid A \in V'\}$

- T'' extended from T' by adding, for each place $p \in P'$, set of transitions $T_{p,\Sigma'}$ and $T_{p,V'}$, i. e., $T'' = T' \cup T_{p,\Sigma'} \cup T_{p,V'}$. For any firing mode ρ, the transitions $t_{p,a}$ and $t_{p,A}$ remove the remaining tokens from the places.
- Each $t_{p,a}$ and $t_{p,A}$, $p \in P'$, has a single input place p and does not have any output places, i. e., $\xi_1''(t_{p,a}) = \xi_1''(t_{p,A}) = \{p\}$ for all $p \in P'$ and $\xi_2''(t_{p,a}) = \xi_2''(t_{p,A}) = \phi$.
- $\gamma''(t_{p,a}) = T_a \to T_a$ and $\gamma''(t_{p,A}) = T_A \to T_A$, $p \in P$.
- M'' consists of the empty marking ϕ, i. e., $M'' = \{\phi\}$.

As we mentioned in the proof of the Lemma 4.2, if in derivations, where used the productions $S \to \lambda$, we cannot replace λ-transitions by transitions labeled $S \to S$ (or $T_a \to T_a$). Therefore, in this case we remove the production $S \to \lambda$ from the original grammar, then apply the above construction. If the original grammar can generate λ, then constructed grammar can be easily modified by adding λ to the language. □

The next theorem illustrates that for any t-type concurrent grammar with erasing rules always, there is its "canonical" form whose control labeled Petri net has one token in the initial marking and no tokens in the final marking.

Theorem 4.6. Every language $L \in \mathbf{PN}^{\lambda}(x, t, \rho)$, $x \in \{-\lambda, \lambda\}$, $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$ can be generated by an (x, t, ρ)-concurrent grammar $\mathcal{G} = (V, \Sigma, S, R, K)$ with $K = (R, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ such that $|\mu_0| = 1$ and $M = \{\phi\}$.

Proof. Let $\mathcal{G} = (V', \Sigma, S', R', K')$ with $K' = (R', P', T', \xi_1', \xi_2', \gamma', \mu_0', M')$ be an (x, t, ρ)-concurrent grammar generating the language L. Let $M' = \{\mu_1, \mu_2, \ldots, \mu_k\}$, $k \geq 1$. We define the sets of the new elements

- $V_S = \{S\} \cup \{X_i \mid 1 \leq i \leq k\}$,
- $R_S = \{S \to S'X_i \mid 1 \leq i \leq k\}$ and $R_M = \{X_i \to \lambda \mid 1 \leq i \leq k\}$,
- $T_S = \{t_{0,i} \mid 1 \leq i \leq k\}$ and $T_M = \{t_{1,i} \mid 1 \leq i \leq k\}$,

and set the components of the equivalent grammar $\mathcal{G} = (V, \Sigma, S, R, K)$ with $K = (R, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ as follows:

- $P = P' \cup \{p_0\}$ and $T = T' \cup T_S \cup T_M$,
- for all $t \in T$,

$$\xi_1(t) = \begin{cases} \xi_1'(t) & \text{if } t \in T', \\ \{p_0\} & \text{if } t \in T_S, \\ \mu_i & \text{if } t = t_{1,i} \in T_M, 1 \leq i \leq k, \end{cases}$$

- for all $t \in T$,

$$\xi_2(t) = \begin{cases} \xi_2'(t) & \text{if } t \in T', \\ \mu_0 & \text{if } t \in T_S, \\ \{\phi\} & \text{if } t \in T_M, \end{cases}$$

- for all $t \in T$,

$$\gamma(t) = \begin{cases} \gamma'(t) & \text{if } t \in T', \\ S \to S'X_i & \text{if } t = t_{0,i} \in T_S, 1 \le i \le k, \\ X_i \to \lambda & \text{if } t = t_{1,i} \in T_M, 1 \le i \le k, \end{cases}$$

- $\mu_0(p_0) = 1$ and $\mu_0(p) = 0$ for all $p \in P - \{p_0\}$,
- $M = \{\phi\}$, i. e., $\phi(p) = 0$ for all $p \in P$.

□

4.3. Comparison of concurrently controlled grammars with Petri net controlled grammars

In this subsection, we study relation between the family of languages generated by concurrent grammars and the family of languages generated by Petri net grammars. From the definition, the following inclusions follow immediately.

Lemma 4.7. For all $x \in \{-\lambda, \lambda\}$, $y \in \{r, t\}$ and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$,

$$\mathbf{PN}(x, y, \rho) \subseteq \mathbf{PN}^{\lambda}(x, y, \rho).$$

Next lemma shows the relation between Petri net grammars and concurrent grammars.

Lemma 4.8. For all $x \in \{-\lambda, \lambda\}$, $y \in \{r, t\}$ and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$,

$$\mathbf{PN}^{[\lambda]}(x, y) \subseteq \mathbf{PN}^{[\lambda]}(x, y, \rho).$$

P r o o f. Let $\mathcal{G} = (V, \Sigma, S, R, K)$ with $K = (R, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ be an (x, y)-Petri net grammar (with or without erasing rules), where $x \in \{-\lambda, \lambda\}$ and $y \in \{r, t\}$. We construct an equivalent (x, y, ρ)-concurrent grammar $\mathcal{G}' = (V, \Sigma, S, R, K')$ with $K = (R, P', T, \xi_1', \xi_2', \gamma, \mu_0', M')$ by adding a new place p_0 with incoming and outgoing edges from and to each transition of T, where $M' = P'^{\oplus}$. The initial and each final marking is extended with by one token in p_0. This place controls that at a time, only one transition of K' can be firable in any firing mode $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$. Formally, $P' = P \cup \{p_0\}$, for all $t \in T$, $\xi_i'(t) = \xi_i(t) \oplus \{p_0\}$, $i = 1, 2$, for all $p \in P$, $\mu_0'(p) = \mu_0(p)$ and $\mu_0'(p_0) = 1$, for each $\mu \in M$, $\nu_\mu \in M'$ is defined as $\nu_\mu(p) = \mu(p)$ and $\nu_\mu(p_0) = 1$. Thus, $L(\mathcal{G}) = L_\rho(\mathcal{G}')$. □

From the Lemma 4.8 and Theorem 2.6, we obtain the lower bound for the families of concurrent languages:

Corollary 4.9. For all $x \in \{-\lambda, \lambda\}$, $y \in \{r, t\}$ and $\rho \in \{m, s, \widehat{s}, \widehat{m}\}$,

$$\mathbf{MAT}^{[\lambda]} \subseteq \mathbf{PN}^{[\lambda]}(x, y, \rho).$$

The following lemma shows that the family of languages generated by concurrent grammars in step mode coincide with the family of languages generated by Petri net grammars.

Lemma 4.10. For all $x \in \{-\lambda, \lambda\}$ and $y \in \{r, t\}$,

$$\mathbf{PN}^{[\lambda]}(x, y, s) = \mathbf{PN}^{[\lambda]}(x, y).$$

Proof. Let $\mathcal{G} = (V, \Sigma, S, R, K)$ with $K = (R, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ be an (x, y, s)-concurrent grammar (with or without erasing rules) where $x \in \{-\lambda, \lambda\}$ and $y \in \{r, t\}$. In an equivalent Petri net grammar $\mathcal{G}'$, the control Petri net K' is expected to simulate every possible multiset $\tau \subseteq T_x$, $x \in R \cup \{\lambda\}$, with a new single transition t_τ with the same label x. Let $\mathcal{G}' = (V, \Sigma, S, R, K')$ with $K' = (R, P, T', \xi_1', \xi_2', \gamma', \mu_0, M)$. We set

$$T' = T \cup \{t_\tau \mid \text{ for all } \tau \subseteq T_x \text{ and for all } x \in R \cup \{\lambda\}\},$$

and we also define

- for all $t \in T$, $\xi_i'(t) = \xi_i(t)$, and the multisets

$$\xi_i'(t_\tau) = \bigoplus_{t \in \tau} \xi_i(t) \text{ and } x \in R \cup \{\lambda\}\}$$

  where $i = 1, 2$,

- for all $t \in T$, $\gamma'(t) = \gamma(t)$ and, for all $\tau \subseteq T_x$, $x \in R \cup \{\lambda\}$, $\gamma'(t_\tau) = x$.

Further, it is not difficult to see that $L_s(\mathcal{G}) = L(\mathcal{G}')$. □

4.4. Comparison of the various firing modes

In this subsection we compare the various firing modes. By using the fact $\mathcal{L}_s^{[\lambda]} \subseteq \mathcal{L}_{\widehat{s}}^{[\lambda]}$ and $\mathcal{L}_m^{[\lambda]} \subseteq \mathcal{L}_{\widehat{m}}^{[\lambda]}$, which is mentioned in Theorem 2.7, one can show that

Lemma 4.11. For all $x \in \{-\lambda, \lambda\}$ and $y \in \{r, t\}$,

$$\mathbf{PN}^{[\lambda]}(x, y, s) \subseteq \mathbf{PN}^{[\lambda]}(x, y, \widehat{s}) \text{ and } \mathbf{PN}^{[\lambda]}(x, y, m) \subseteq \mathbf{PN}^{[\lambda]}(x, y, \widehat{m}).$$

The next lemma shows that concurrent grammars in step and maximal step modes can be simulated by concurrent grammars in multistep and maximal multistep modes, respectively.

This lemma follows directly from the fact $\mathcal{L}_s^{[\lambda]} \subseteq \mathcal{L}_m^{[\lambda]}$ and $\mathcal{L}_{\widehat{s}}^{[\lambda]} \subseteq \mathcal{L}_{\widehat{m}}^{[\lambda]}$, which is mentioned in Theorem 2.7.

Lemma 4.12. For all $x \in \{-\lambda, \lambda\}$ and $y \in \{r, t\}$,

$$\mathbf{PN}^{[\lambda]}(x, y, s) \subseteq \mathbf{PN}^{[\lambda]}(x, y, m) \text{ and } \mathbf{PN}^{[\lambda]}(x, y, \widehat{s}) \subseteq \mathbf{PN}^{[\lambda]}(x, y, \widehat{m}).$$

4.5. Comparison of concurrently controlled grammars with Petri nets with labeled steps

The following lemma shows that all Petri net languages of all variants of multisteps can be generated by t-type concurrent grammars with erasing rules.

Lemma 4.13. For $\rho \in \{s, m, \widehat{s}, \widehat{m}\}$,

$$\mathcal{L}_{\rho}^{[\lambda]} \subseteq \mathbf{PN}^{\lambda}(\lambda, t, \rho).$$

Proof. Let $K = (\Sigma, P, T, \xi_1, \xi_2, \gamma, \mu_0, M)$ be a labeled Petri net where M is a finite set. For $\rho \in \{s, m, \widehat{s}, \widehat{m}\}$, we construct the (λ, t, ρ)-concurrent grammar $\mathcal{G} = (V, \Sigma, S, R, K')$ with $K' = (R, P', T, \xi_1', \xi_2', \gamma', \mu_0', M')$ such that $L_{\rho}(K) = L_{\rho}(\mathcal{G})$. We construct K' from K by adding two new places p_0 and p_1, two transitions t_0 and t_1. We also add the arc from p_0 to t_0, from p_1 to t_1, and from t_0 to p_1 and the arcs with the weights $\mu_0(p)$ to the places p of K where $\mu_0(p) > 0$. The labels of t_0 and t_1 are $S \to S$ and $S \to \lambda$, respectively. Each label $x \in \Sigma \cup \{\lambda\}$ assigned to a transition of K is replaced with the label $S \to xS$ in K'. The initial marking μ_0' has only one token in p_0. Each final marking in M' is defined from some final marking in M, which has the same number of tokens in the places of K, and additionally, there are no tokens in p_0 and p_1. It is not difficult to observe that for any prefix u of a terminal string of K, there is a sentential form uS generated by $\mathcal{G}$ and vice versa. On the one hand, we can obtain the terminal string w in $L(\mathcal{G})$ if reach some final marking in K, where we have the sentential form wS, and then erase S, which results in a final marking of K'. The string w is also a string of $L(K)$. On the other hand, every terminal string of $L(K)$ can be easily obtained by direct simulation it in $\mathcal{G}$. Thus, $L_{\rho}(K) = L_{\rho}(\mathcal{G})$. □

Combining the results of Lemma 4.13 and Theorem 6 in [21], where it was shown $\mathcal{L}_{\widehat{s}}^{\lambda} = \mathcal{L}_{\widehat{m}}^{\lambda} = \mathbf{RE}$, we obtain

Theorem 4.14.

$$\mathbf{CS} \subset \mathbf{PN}^{\lambda}(\lambda, t, \widehat{s}) = \mathbf{PN}^{\lambda}(\lambda, t, \widehat{m}) = \mathbf{RE}.$$

5. CONCLUSION

In this paper, we have proposed new variants of the parallel computation using p/t Petri nets under parallel firing strategies, called *concurrently controlled grammars*, which are natural formal models for concurrent, asynchronous, distributed, parallel, nondeterministic and stochastic systems.

We have defined concurrently controlled grammars in step, multistep, maximal step, maximal multistep modes, with arbitrary and λ-free labeling, and r- and t-type final markings. We have investigated the computational power of the introduced concurrently controlled grammars. Figure 1 illustrates the relations of the families of languages generated by Petri net controlled grammars under sequential and parallel firing strategies.

Since this work is a preliminary research, relationships among the language families have not been established completely. For instance, the relationship between

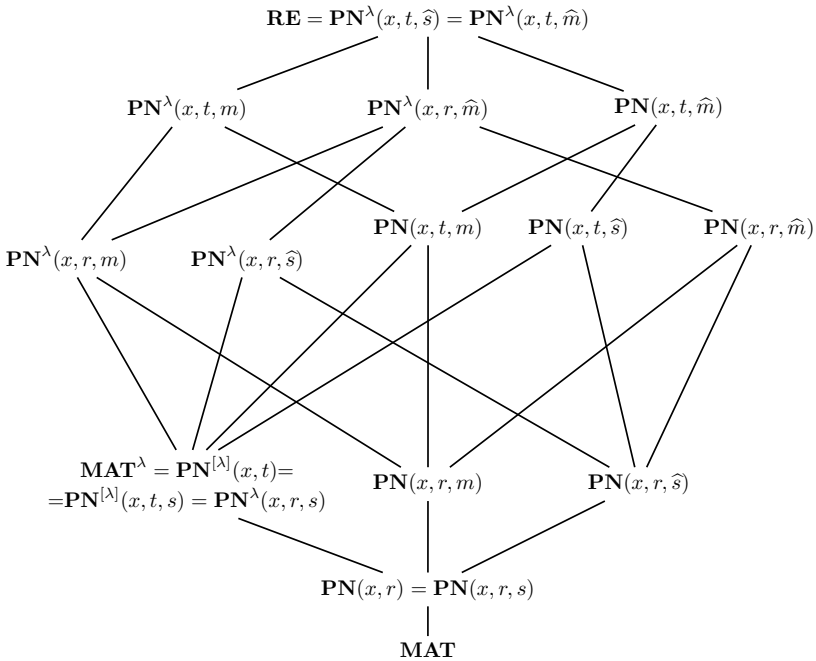

Fig. 1. The hierarchy of language families generated by (sequential and concurrent) Petri net controlled grammars (the lines denote inclusions of the lower families into the upper families).

$\mathbf{PN}^{[\lambda]}(x,y,\widehat{s})$ and $\mathbf{PN}^{[\lambda]}(x,y,m)$ is unclear and it needs further investigations. The strictness of the inclusions

$$\mathbf{PN}(x,y,\rho) \subseteq \mathbf{PN}^{\lambda}(x,y,\rho) \text{ and } \mathbf{PN}^{[\lambda]}(x,y) \subseteq \mathbf{PN}^{[\lambda]}(x,y,\rho)$$

remains open, as well. Further, it remains to investigate concurrently controlled grammars in nonterminal labeling mode, with free labeling strategy and other definitions of final marking sets.

ACKNOWLEDGMENTS

The first, second and fourth authors would like to thank Universiti Putra Malaysia for the financial funding through **FRGS/1/2014/ICT03/UPM/01/1**. The third author is grateful to the International Islamic University Malaysia for his financial funding of Research Initiative Grant Scheme **RIGS16-368-0532**. The fifth and sixth authors are grateful to Eurasian National University for the financial funding through Research Grant Scheme.

(Received August 23, 2016)

REFERENCES

[1] M. Beek and H. Kleijn: Petri net control for grammar systems. In: Formal and Natural Computing, volume 2300 of LNCS, Springer 2002, pp. 220–243. DOI:10.1007/3-540-45711-9_13

[2] H.-D. Burkhard: Ordered firing in petri nets. Inform. Process. Cybernet. *2* (1989), 3, 71–86.

[3] J. Dassow, G. Mavlankulov, M. Othman, S. Turaev, M. Selamat, and R. Stiebe: Grammars controlled by petri nets. In: Petri Nets – Manufacturing and Computer Science, InTech 2012, pp. 337–358. DOI:10.5772/50637

[4] J. Dassow and S. Turaev: Arbitrary Petri net controlled grammars. In: Linguistics and Formal Languages (G. Bel-Enguix and M. Jiménez-López, eds.), Second International Workshop on Non-Classical Formal Languages In Linguistics, Tarragona 2008, pp. 27–39.

[5] J. Dassow and S. Turaev: k-Petri net controlled grammars. In: Pre-proceedings of Second International Conference on Language and Automata Theory and Applications. Report 36/08, Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona 2008.

[6] J. Dassow and S. Turaev: k-Petri net controlled grammars. In: Language and Automata Theory and Applications (C. Martín-Vide, F. Otto, and H. Fernau, eds.), Second International Conference, LATA 2008. Revised Papers, volume 5196 of LNCS, Springer 2008, pp. 209–220. DOI:10.1007/978-3-540-88282-4_20

[7] J. Dassow and S. Turaev: Grammars controlled by special Petri nets. In: Language and Automata Theory and Applications (A. Dediu, A.-M. Ionescu, and C. Martín-Vide, eds.), Third International Conference, LATA 2009, volume 5457 of LNCS, Springer 2009, pp. 326–337. DOI:10.1007/978-3-642-00982-2_28

[8] J. Dassow and S. Turaev: Petri net controlled grammars: the case of special Petri nets. J. Universal Computer Sci. *15* (2009), 14, 2808–2835.

[9] J. Dassow and S. Turaev: Petri net controlled grammars: the power of labeling and final markings. Romanian J. Inform. Sci. Technol. *12* (2009), 2, 191–207.

[10] J. Dassow and S. Turaev: Petri net controlled grammars with a bounded number of additional places. Acta Cybernet. *19* (2010), 609–634.

[11] B. Farwer, M. Jantzen, M. Kudlek, H. Rölke, and G. Zetzsche: Petri net controlled finite automata. Fundamenta Inform. *85* (2008), 1–4, 111–121.

[12] B. Farwer, M. Kudlek, and H. Rölke: Petri-net-controlled Machine Models. Technical Report 274, FBI-Bericht, Hamburg 2006.

[13] B. Farwer, M. Kudlek, and H. Rölke: Concurrent Turing machines. Fundamenta Inform. *79* (2007), 3–4, 303–317.

[14] A. Ginzburg and M. Yoeli: Vector addition systems and regular languages. J. Comput. System Sci. *20* (1980), 227–284. DOI:10.1016/0022-0000(80)90009-4

[15] M. Hack: Petri Net Languages. Computation Structures Group Memo, Project MAC 124, MIT, 1975.

[16] D. Hauschildt and M. Jantzen: Petri net algorithms in the theory of matrix grammars. Acta Inform. *31* (1994), 719–728. DOI:10.1007/bf01178731

[17] M. Jantzen: On the hierarchy of Petri net languages. RAIRO Theoret. Inform. *13* (1979), 1, 19–30. DOI:10.1051/ita/1979130100191

[18] M. Jantzen: Language theory of Petri nets. In: Advances in Petri Nets, volume 254 of LNCS, Springer 1987, pp. 397–412.

[19] M. Jantzen, M. Kudlek, and G. Zetzsche: Language classes defined by concurrent finite automata. Fundamenta Inform. *85* (2008), 1–4, 267–280.

[20] M. Jantzen and H. Petersen: Cancellation in context-free languages: Enrichment by induction. Theoret. Computer Sci. *127* (1994), 149–170. DOI:10.1016/0304-3975(94)90104-x

[21] M. Jantzen and G. Zetzsche: Labeled step sequences in Petri nets. In: Applications and Theory of Petri Nets, Proc. 29th International Conference, PETRI NETS 2008, volume 5062, pp. 270–287. DOI:10.1007/978-3-540-68746-7_19

[22] V. Marek and M. Češka: Petri nets and random-context grammars. In: Proc. 35th Spring Conference: Modelling and Simulation of Systems, MARQ Ostrava, Hradec nad Moravicí 2001, pp. 145–152.

[23] G. Mavlankulov, M. Othman, M. H. Selamat, and S. Turaev: Concurrent context-free grammars. In: Proc. First International Conference on Advanced Data and Information Engineering (DaEng-2013), LNEE, pp. 521–528. DOI:10.1007/978-981-4585-18-7_58

[24] G. Mavlankulov, M. Othman, M. H. Selamat, and S. Turaev: Some properties of the concurrent grammars. In: International Conference on Mathematical Sciences and Statistics 2013, Springer Singapore 2014, pp. 223–231. DOI:10.1007/978-981-4585-33-0_23

[25] G. Mavlankulov, S. Turaev, L. Zhumabaeva, and T. Zhukabayeva: Parallel firing strategy on petri nets: A review. In: International Conference On Mathematics, Engineering And Industrial Applications 2014 (ICoMEIA 2014), volume 1660, AIP Publishing, 2015, pp. 5–8. DOI:10.1063/1.4915713

[26] C. Petri: Kommunication mit Automaten. PhD Thesis, University of Bonn 1962.

[27] M. Selamat and S. Turaev: Grammars controlled by Petri nets with place capacities. In: 2010 International Conference on Computer Research and Development 2010, pp. 51–55.

[28] R. Stiebe and S. Turaev: Capacity bounded grammars. J. Automata, Languages Combinatorics *15* (2009), 1/2, 175–194.

[29] R. Stiebe and S. Turaev: Capacity bounded grammars and Petri nets. EPTCS *3* (2009), 193–203. DOI:10.4204/eptcs.3.18

[30] R. Stiebe and S. Turaev: Capacity bounded grammars and Petri nets. In: 11th International Workshop on Descriptional Complexity of Formal Systems (J. Dassow, G. Pighizzini, and B. Truthe, eds.), Magdeburg 2009, pp. 247–258.

[31] S. Turaev: Semi-matrix grammars. In: Second Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2006, Mikulov, pp. 245–252.

[32] S. Turaev: Petri net controlled grammars. In Third Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2007, Mikulov, pp. 233–240.

[33] S. Turaev, A. Krassovitskiy, M. Othman, and M. Selamat: Parsing algorithms for grammars with regulated rewriting. In: Recent Researches in Applied Informatics and Remote Sensing, 11th WSEAS International Conference on Applied Computer Science 2011 (A. Zaharim, K. Sopian, N. Mostorakis, and V. Mladenov, eds.), pp. 103–109.

[34] S. Turaev, A. Krassovitskiy, M. Othman, and M. Selamat: Parsing algorithms for regulated grammars. Math. Models Methods Appl. Sci. *6* (2012), 6, 748–756.

[35] R. Valk and G. Vidal-Naquet: Petri nets and regular languages. J. Computer System Sci. *23* (2981), 23, 229–325. DOI:10.1016/0022-0000(81)90067-2

[36] H. Yen: On the regularity of petri net languages. Inform. Comput. *124* (1996), 168–181. DOI:10.1006/inco.1996.0013

[37] G. Zetzsche: Erasing in petri net languages and matrix grammars. In: Proc. 13th International Conference on Developments in Language Theory, DLT '09, Berlin, Heidelberg 2009, pp. 490–501. DOI:10.1007/978-3-642-02737-6_40

[38] G. Zetzsche: A sufficient condition for erasing productions to be avoidable. In: Developments in Language Theory, 15th International Conference, DLT 2011, Milan 2011, volume 6795 LNCS, Springer 2011, pp. 452–463. DOI:10.1007/978-3-642-22321-1_39

Gairatzhan Mavlankulov, Department of Communication Technology and Network, Universiti Putra Malaysia,43400, UPM Serdang, Selangor D.E. Malaysia.
e-mail: gairatjon@gmail.com

Mohamed Othman, Department of Communication Technology and Network, Universiti Putra Malaysia,43400,UPM Serdang, Selangor D.E. Malaysia.
e-mail: mothman@upm.edu.my; mothman@ieee.org

Sherzod Turaev, Department of Computer Science, International Islamic University Malaysia, 53100, Gombak, Selangor D.E. Malaysia.
e-mail: sherzod@iium.edu.my

Mohd Hasan Selamat, Department of Information Systems, Universiti Putra Malaysia, 43400, UPM Serdang, Selangor D.E. Malaysia.
e-mail: hasan@upm.edu.my

Laula Zhumabayeva, International Department, S.Yessenov Caspian State University of Technologies and Engineering, 130003, Aktau. Kazakhstan.
e-mail: lau_la@mail.ru

Tamara Zhukabayeva, Department of Information Technologies, L.N. Gumilyov Eurasian National University, 010008, Astana. Kazakhstan.
e-mail: tamara.kokenovna@gmail.com