Acta Cybernetica **14** (1999) 193-203.

Generalized fairness and context-free languages

Kai Salomaa* Sheng Yu*†

Abstract

The notion of fairness for generalized shuffle operations was introduced in [10]. The n-fairness property requires, roughly speaking, that in any prefix of a word the difference of the numbers of occurrences of two symbols is at most n. Here we give a new simplified proof for the decidability of uniform n-fairness for context-free languages. Also, we show that the more general, linear or logarithmic, fairness notions are decidable.

1 Introduction

Fairness constraints can be used to restrict the behavior of concurrent processes. For a state sequence to be fair, a minimal requirement is that a process that is enabled infinitely often will occur infinitely often. Many different notions of fairness have been studied in modeling concurrency [3], some recent references are [2, 5, 14]. Fairness of automata on infinite objects and of cooperating grammar systems is considered in [8, 13, 15].

Questions of fairness in formal language theory were initiated by the study of trajectories [9, 10]. A trajectory is a word over a two-letter alphabet that is used to "control" the shuffle operation on given words. Trajectories yield very general operations of parallel composition of words and languages and have applications in the parallelization of languages, in representing a language in terms of simpler components.

A trajectory $t \in \{b, c\}^*$ is said to be n-fair if the difference in the number of occurrences of the symbols b and c in any prefix of t is at most n. The shuffle of two words that is controlled by an n-fair trajectory satisfies thus the property that at any point neither word is more than "n steps ahead". It is easy to see that for a context-free set of trajectories T we can effectively decide whether or not T is n-fair. The uniform fairness question asks, for a given set of trajectories T, whether or not there exists an integer n such that T is n-fair.

*Department of Computer Science, University of Western Ontario, London, Ontario N6A 5B7, Canada, Email: {ksalomaa, syu}@csd.uwo.ca

†Research supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0041630.

It was shown in [11] that also the uniform fairness property is decidable for context-free languages. The proof establishes that for a given context-free grammar G there exists a constant m_G such that the uniform fairness of $L(G)$ can be verified by checking only the derivation trees of height at most m_G. The resulting algorithm is not efficient since it uses an exhaustive search of an exponential number of derivation trees (in terms of the size of the grammar). Furthermore, the argument used to establish the existence of the constant m_G uses fairly complicated operations on the derivation trees.

Here we give a new proof for the decidability of uniform fairness for context-free languages. The fairness property for a context-free language L is determined by considering a regular language that is letter-equivalent to the prefix-language of L and thus it is, essentially, sufficient to decide the property for a regular language. The argument for the correctness of the algorithm is much simpler than in the original proof that relies directly on properties of context-free derivations. Here the adjective "simpler" naturally ignores the fact that we are using the powerful result of Parikh's theorem [12, 4, 16]. Another drawback is that the size of the nondeterministic finite automaton (or regular grammar) obtained by Parikh's theorem to accept a language letter-equivalent to the given context-free language can be much larger than the size of the original context-free grammar. Furthermore, although the algorithm is now conceptually simpler, in the worst case it needs to check an exponential number of cycles in the automaton.

An advantage of the new decidability proof is that exactly the same method allows us to decide the generalized (linear or logarithmic) fairness conditions for context-free languages that were left open in [10].

2 Definitions

Here we present some definitions needed later. More details on formal languages and finite automata can be found e.g. in [1, 4, 16, 17]. For all unexplained notions we refer the reader to these references.

The symbol $\mathbb{N}$ denotes the set of non-negative integers. The cardinality of a finite set S is denoted $\#S$. The set of words over an alphabet Σ is Σ^* and $\Sigma^+ = \Sigma^* - \{\lambda\}$. Here λ denotes the empty word. If not otherwise mentioned, by an alphabet we mean always a finite alphabet. A word w_1 is a *prefix* of $w \in \Sigma^*$ if we can write $w = w_1 w_2$, $(w_1, w_2 \in \Sigma^*)$. For $L \subseteq \Sigma^*$, the prefix-language of L is defined as

$$\mathrm{pref}(L) = \{w \in \Sigma^* \mid (\exists w' \in \Sigma^*)\ ww' \in L\}.$$

The length of a word $w \in \Sigma^*$ is $|w|$ and, for a symbol $c \in \Sigma$, $|w|_c$ denotes the number of occurrences of c in the word w.

Words $w_1, w_2 \in \Sigma^*$ are said to be *letter-equivalent* if for each $c \in \Sigma$ we have $|w_1|_c = |w_2|_c$. Languages L_1 and L_2 $(\subseteq \Sigma^*)$ are *letter-equivalent* if for each $w_1 \in L_1$ there exists $w_2 \in L_2$ such that w_1 and w_2 are letter-equivalent, and vice versa. This means that the words of L_2 are exactly some permutations of the words of L_1.

A *finite automaton* is a four-tuple $A = (Q, \Sigma, s, Q', \delta)$ where Q is the finite set of states, Σ is the input alphabet, $s \in Q$ is the initial state, $Q' \subseteq Q$ is the set of accepting final states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation. Note that for given $q \in Q$ and $a \in \Sigma$ there may exist more than one state q' such that $(q, a, q') \in \delta$, that is, the automaton is allowed to be nondeterministic.

The transition relation is extended in the natural way from symbols of Σ to arbitrary words of Σ^* and we denote also the extended transtion relation ($\subseteq Q \times \Sigma^* \times Q$) by the same symbol δ. The *language accepted by* A is

$$L(A) = \{w \in \Sigma^* \mid (\exists q \in Q')\ (s, w, q) \in \delta\}.$$

A *path* of the automaton A is a sequence

$$\alpha = (q_1, a_1, q_2, a_2, \ldots, a_{m-1}, q_m) \tag{1}$$

where $m \geq 1$, $q_i \in Q$, $i = 1, \ldots, m$, $a_j \in \Sigma$, $j = 1, \ldots, m-1$, and $(q_j, a_j, q_{j+1}) \in \delta$ for all $j \in \{1, \ldots, m-1\}$.

The above path is said to be *accepting* if $q_1 = s$ and $q_m \in Q'$. The automaton A is said to be *reduced* if all states of Q occur in some accepting path of A. It is well known that we can determine the unnecessary states of an automaton and, thus, we can effectively transform A into an equivalent reduced automaton. The *underlying word* of a path α as in (1) is

$$\mathrm{word}(\alpha) = a_1 a_2 \cdots a_{m-1} \in \Sigma^*.$$

A path (1) is said to be a *cycle* if $m \geq 2$ and $q_1 = q_m$. Note that a sequence (q_1) consisting of a single state (with no transitions) is a path but not a cycle. A cycle as in (1) is said to be *primitive* if for no $(i, j) \neq (1, m)$, $1 \leq i < j \leq m$, the sequence

$$(q_i, a_i, q_{i+1}, a_{i+1}, \ldots, a_{j-1}, q_j) \tag{2}$$

is a cycle. A path as in (1) is said to be primitive, if for no $1 \leq i < j \leq m$, the sequence (2) is a cycle.

Intuitively, a primitive cycle does not contain any proper subcycles and a primitive path does not contain any subcycles. In (1) the q_i's need not be distinct and, thus, an automaton A may have an infinite number of cycles (or paths). However, the number of primitive cycles and paths is always finite.

We can define in the natural way the catenation of two paths provided that the first one "ends" with the same state as the second one "begins" with. Let α be as in (1) and

$$\beta = (p_1, b_1, \ldots, b_{r-1}, p_r),$$

where $p_i \in Q$, $b_j \in \Sigma$, $1 \leq i \leq r$, $1 \leq j \leq r-1$. If $q_m = p_1$ then the *catenation* of the paths α and β is defined to be

$$\alpha \cdot \beta = (q_1, a_1, q_2, a_2, \ldots, a_{m-1}, q_m, b_1, p_2, \ldots, b_{r-1}, p_r).$$

If $q_m \neq p_1$ then the catenation of α and β is not defined. Note that a cycle can always be catenated with itself.

Finally we fix the notation used for context-free grammars. A *context-free grammar* is a four-tuple $G = (N, \Sigma, S, P)$, where N is the nonterminal alphabet, Σ is the terminal alphabet, $(N \cap \Sigma = \emptyset)$, $S \in N$ is the initial nonterminal, and $P \subseteq N \times (N \cup \Sigma)^*$ is the finite set of productions. A production $(X, w) \in P$ is denoted as $X \to w$. The productions define in the standard way the rewrite relation of the grammar $\Rightarrow_G \subseteq (N \cup \Sigma)^* \times (N \cup \Sigma)^*$, and the language generated by G is $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

A context-free grammar $G = (N, \Sigma, S, P)$ is in *Chomsky normal form* if all productions of P are of the form $X \to YZ$ or $X \to b$ where $X, Y, Z \in N$ and $b \in \Sigma$. The grammar G is said to be *regular* (or right-linear) if all productions of P are of the form $X \to wY$ or $X \to w$ where X and Y are nonterminals and w is a terminal string. Every context-free language not containing the empty word can be generated by a grammar in Chomsky normal form. The regular grammars generate exactly the regular languages.

When speaking about the complexity of determining some property of context-free grammars, by the size of the grammar we mean the length of an encoding over a fixed (e.g. a binary) alphabet of the nonterminals, the terminals and the productions of the grammar. Similarly, the size of a finite automaton is determined by an encoding of the states, the input alphabet and the transition relation.

3 Generalized fairness

The n-fairness condition [10, 11] requires that for any distinct symbols b and c and any prefix w' of a given word w, the difference between the numbers of occurrences of b and c in w' is bounded by n. A more general notion of fairness was also informally discussed in [11]. Below we present the more general definition.

Definition 3.1 *Let $g : \mathbb{N} \to \mathbb{N}$ be a function. We say that a language $L \subseteq \Sigma^*$ has the g-fairness property if the following condition holds. For all $b, c \in \Sigma$, if $w = w_1 w_2 \in L$ then*

$$| \, |w_1|_b - |w_1|_c \, | \leq g(|w_1|).$$

The definition requires that the difference between the numbers of occurrences of distinct symbols in any prefix of a word belonging to the language is bounded by the g-image of the length of the prefix. As a special case we get the notion of n-fairness, $n \in \mathbb{N}$, by choosing g to be the function with constant value n.

When we are using a fairness condition as given in Definition 3.1, the function g is referred to as the *fairness function* associated with the condition.

It is a straightforward observation that given a context-free language L and $n \in \mathbb{N}$ the question whether or not L is n-fair is decidable [10]. The *uniform constant-fairness question* asks for a given language L whether or not there exists $n \in \mathbb{N}$ such that L is n-fair. The following result was shown in [11].

Theorem 3.1 *The uniform constant-fairness problem is decidable for context-free languages.*

Below we give a new simplified proof for Theorem 3.1. We will use the following two propositions. A proof of Parikh's theorem can be found for instance in [12, 4, 16]. A more elegant proof using equations over a commutative semigroup is presented in [1, 7].

Proposition 3.1 *(Parikh's Theorem) Each context-free language is letter-equivalent to a regular language. Given a context-free grammar we can effectively construct a regular grammar (or finite automaton) for a letter-equivalent regular language.*

Proposition 3.2 *The prefix-language of a context-free language is context-free.*

Proposition 3.2 follows from the observation that if a language L is generated by a grammar $G = (N, \Sigma, S, P)$ in Chomsky normal form (with possibly an additional production $S \to \lambda$) then $\mathrm{pref}(L)$ is generated by the grammar G' defined as follows. Denote $N' = \{X' \mid X \in N\}$ and let $G' = (N \cup N', \Sigma, S', P \cup P')$ where

$$\begin{aligned} P' \;=\; & \{X' \to YZ',\ X' \to Y' \mid X \to YZ \in P;\ X, Y, Z \in N\} \\ & \cup \{X' \to b \mid X \to b \in P;\ X \in N,\ b \in \Sigma\} \cup \{S' \to \lambda\}. \end{aligned}$$

We assume that S, and hence also S', does not appear in the right side of any production.

Now we can prove Theorem 3.1 relying on the above results.

Proof of Theorem 3.1. Let G be a given context-free grammar with terminal alphabet Σ. By Proposition 3.2 there exists effectively a context-free grammar G' such that $L(G') = \mathrm{pref}(L(G))$. Now there exists $n \in \mathbb{N}$ such that $L(G)$ has the n-fairness property iff there exists $n \in \mathbb{N}$ such that for all $w \in L(G')$,

$$(\forall b, c \in \Sigma) \ \mid |w|_b - |w|_c| \leq n. \tag{3}$$

By Parikh's theorem there exists effectively a finite automaton A such that $L(A)$ is letter-equivalent to $L(G')$. This means that, given $n \in \mathbb{N}$, the condition (3) holds for all $w \in L(G')$ iff the same condition holds for all $w \in L(A)$.

Without loss of generality we can assume that A is reduced. Let $n \in \mathbb{N}$ be fixed. We claim that the condition (3) holds for all $w \in L(A)$ iff the below conditions (i) and (ii) hold.

(i) For all accepting primitive paths α of A, and all $b, c \in \Sigma$,

$$\mid |\mathrm{word}(\alpha)|_b - |\mathrm{word}(\alpha)|_c \mid \leq n.$$

(ii) For all primitive cycles β of A, and all $b, c \in \Sigma$,

$$|\mathrm{word}(\beta)|_b = |\mathrm{word}(\beta)|_c.$$

For the "only if" part assume that (3) holds for all $w \in L(A)$. Consider arbitrary $b, c \in \Sigma$. Now (i) follows from the fact that word$(\alpha) \in L(A)$ if α is an accepting path. For the sake of contradiction assume that

$$\beta = (q_1, a_1, \ldots, q_{m-1}, a_{m-1}, q_1),$$

$q_i \in Q$, $a_j \in \Sigma$, $1 \leq i, j \leq m-1$, $m \geq 2$, is a (primitive) cycle where, for instance,

$$|\text{word}(\beta)|_b > |\text{word}(\beta)|_c.$$

Since A is reduced, the state q_1 is reachable from the initial state and a final state is reachable from q_1. Thus, there exists an accepting path of A that can be written in the form $\gamma_1 \cdot \beta \cdot \gamma_2$. Denote

$$k = |\, |\text{word}(\gamma_1 \cdot \gamma_2)|_b - |\text{word}(\gamma_1 \cdot \gamma_2)|_c \,|.$$

Then

$$\eta = \gamma_1 \cdot \beta^{k+n+1} \cdot \gamma_2$$

is an accepting path such that word(η) does not satisfy (3).

Conversely, assume that the conditions (i) and (ii) hold and let $b, c \in \Sigma$ be arbitrary. Starting from an arbitrary cycle we can delete primitive cycles one-by-one without changing the difference between the numbers of occurrences of the symbols b and c. The process eventually results in a primitive cycle and, thus, it follows that any cycle of A has equally many occurrences of the symbols b and c. An arbitrary accepting path can be written in the form

$$\gamma_1 \cdot \eta_1 \cdot \gamma_2 \cdot \ldots \cdot \eta_{k-1} \cdot \gamma_k,$$

where $k \geq 1$, η_i, $1 \leq i \leq k-1$, is a cycle and $\gamma_1 \cdot \gamma_2 \cdot \ldots \cdot \gamma_k$ is a primitive accepting path. This completes the proof since the words of $L(A)$ are exactly the underlying words of accepting paths of A. □

In the above proof, the answer for a reduced automaton A is "yes" for sufficiently large n iff the condition (ii) holds, that is, the condition (i) would not be needed at all. The condition (i) was included only to make more transparent the idea that the same method can be used below for the linear and logarithmic fairness functions. When considering a fixed fairness function, we need to have some condition also for the accepting primitive paths.

The above proof of Theorem 3.1 is quite simple when compared to the proof given in [11], at least if we ignore the fact that we are relying on Parikh's theorem. On the other hand, the algorithm obtained from the original proof is more efficient, although it also requires exponential time. Note that the construction that for a given context-free grammar G produces a regular grammar generating a language letter-equivalent to $L(G)$ greatly increases the size of the grammar [4, 12, 16]. We do not know whether the construction could be improved in this respect.

The construction using Chomsky normal form grammars outlined above for the proof of Proposition 3.2 also increases the size of a given grammar since the

Chomsky normal form can have many more nonterminals and productions than the original grammar. However, this overhead could be avoided. The construction in the proof of Proposition 3.2 does not require that the grammar is in Chomsky normal form if one uses more carefully defined rules to determine the behavior of the primed nonterminal that denotes the end of the prefix in the derivation tree. Also, the assumption that the given finite automaton is reduced is not problematic since this property can be tested in low polynomial time.

Although it is perhaps intuitively simpler than the construction of [11] that determines properties of context-free derivations, the algorithm testing the primitive loops of a finite automaton still requires exponential time. This follows from the below example.

Example 3.1 *Let $n \in \mathbb{N}$ and consider the finite automaton $A_n = (Q, \Sigma, s, Q', \delta)$ where $\Sigma = \{a, b\}$, $Q = \{1, 2, \ldots, n\}$, $s = 1$, $Q' = \{n\}$, and δ consists of the transitions $(i, x, i+1)$, $(n, x, 1)$ where $i = 1, \ldots, n-1$ and $x \in \{a, b\}$. Then the size of A_n is $O(n \cdot \log n)$ and the number of primitive cycles in A_n is $O(2^n)$.*

By the above remarks, the decision algorithm given by the proof of Theorem 3.1 is extremely inefficient. However, it is useful because with small modifications the same method allows us to show that also the generalized (logarithmic or polynomial) fairness condition is decidable. This question was left open in [11]. First we consider the case where the fairness function is linear.

Theorem 3.2 *Let $g(x) = \tau x + \kappa$ be a linear function where τ and κ are constants. For a given context-free grammar G we can effectively decide whether or not $L(G)$ has the g-fairness property.*

Proof. Let Σ be the terminal alphabet of G. Similarly as in the proof of Theorem 3.1, by Propositions 3.1 and 3.2, it is sufficient to check, for a reduced finite automaton A, and for each pair of symbols $b, c \in \Sigma$, whether or not

$$(\forall w \in L(A)) \;\; | \, |w|_b - |w|_c \, | \leq \tau |w| + \kappa. \tag{4}$$

If τ is negative, we can decide (4) by determining whether $L(A)$ contains a word w that makes $\tau|w| + \kappa$ negative and then going through the finite number of shorter words. Thus, we can assume that τ is non-negative in the following.

We claim that (4) is equivalent to the below two conditions:

(i) For all accepting primitive paths α of A,

$$| \, |\mathrm{word}(\alpha)|_b - |\mathrm{word}(\alpha)|_c \, | \leq \tau |\mathrm{word}(\alpha)| + \kappa.$$

(ii) For all primitive cycles β of A,

$$| \, |\mathrm{word}(\beta)|_b - |\mathrm{word}(\beta)|_c \, | \leq \tau |\mathrm{word}(\beta)|.$$

First assume (4). This directly implies (i) since word(α) $\in L(A)$ for all accepting paths α. If (ii) would not hold then there exists a cycle β such that

$$| \, |\text{word}(\beta)|_b - |\text{word}(\beta)|_c \, | > \tau |\text{word}(\beta)|.$$

Since A is reduced, it follows that A has an accepting path of the form $\gamma_1 \cdot \beta \cdot \gamma_2$. (This is seen using a similar argument as in the proof of Theorem 3.1.) Since word($\gamma_1 \cdot \gamma_2$) $\in L(A)$, we have

$$| \, |\text{word}(\gamma_1 \cdot \gamma_2)|_b - |\text{word}(\gamma_1 \cdot \gamma_2)|_c \, | \leq \tau |\text{word}(\gamma_1 \cdot \gamma_2)| + \kappa.$$

Denote the constant $\tau|\text{word}(\gamma_1 \cdot \gamma_2)|$ by D, and let

$$\eta(m) = \gamma_1 \cdot \beta^m \cdot \gamma_2, \quad m \in \mathbb{N}.$$

Then

$$\text{word}(\eta(2D + 2\kappa + 1)) \in L(A)$$

violates the condition (4).

For the converse part assume the conditions (i) and (ii). Similarly as in the proof of Theorem 3.1, we observe that an arbitrary accepting path of A can be written in the form

$$\gamma_1 \cdot \eta_1 \cdot \gamma_2 \cdot \ldots \cdot \eta_{k-1} \cdot \gamma_k, \tag{5}$$

where $k \geq 1$, η_i, $1 \leq i \leq k-1$, is a cycle and $\gamma_1 \cdot \gamma_2 \cdots \gamma_k$ is a primitive accepting path. The inequality of condition (ii) extends to arbitrary cycles and, thus, the underlying word of (5) has to satisfy (4). □

We can naturally consider also the uniform linear fairness question: Given a context-free grammar G decide whether or not there exist constants τ and κ such that $L(G)$ has the $(\tau x + \kappa)$-fairness property. From the proof of Theorem 3.2 it follows that the answer to this question is "yes" iff all primitive cycles of the constructed automaton A contain occurrences of each symbol of Σ.

As a corollary we see that also the logarithmic and polynomial fairness conditions are decidable for context-free languages. The question of logarithmic or polynomial fairness is reduced, essentially, to the linear case by testing separately a finite number of special cases.

Corollary 3.1 *For a context-free language L it is decidable whether or not L has the* log*-fairness property.*

Proof. Let Σ be the alphabet of L. Exactly as in the first part of the proof of Theorem 3.2, in order to decide the log-fairness property for L, it is sufficient to decide for a reduced finite automaton A, and for all $b, c \in \Sigma$, whether or not

$$(\forall w \in L(A)) \;\; | \, |w|_b - |w|_c \, | \leq \log(|w|). \tag{6}$$

Let $b, c \in \Sigma$ be fixed. We first observe that if for some cycle α of A we have

$$|\text{word}(\alpha)|_b \neq |\text{word}(\alpha)|_c \tag{7}$$

then the condition (6) does not hold. To see this observe that since A is reduced, there exists an accepting path $\gamma_1 \cdot \alpha \cdot \gamma_2$ and, by pumping the cycle α sufficiently many times, we obtain an accepting path such that the corresponding underlying word violates (6).

We can effectively test that no cycle α satisfies (7) just by going through the primitive cycles. If this is the case, then we can determine whether or not (6) holds by checking the finite number of primitive accepting paths. □

Note that the above proof relies only on the fact that the fairness function grows asymptotically slower than any linear function.

Corollary 3.2 *Let $p : \mathbb{N} \to \mathbb{N}$ be a polynomial function. For a context-free language L over an alphabet Σ it is decidable whether or not L has the p-fairness property.*

Proof. Since the linear case was considered above we can assume that the rank of $p(x)$ is at least two. Furthermore, we may assume that the coefficient of the term of highest rank in $p(x)$ is positive because otherwise any infinite language would not have the $p(x)$-fairness property. (Note that we can effectively decide whether or not a given context-free language is infinite.)

Again it is sufficient to decide for a reduced finite automaton A, and for all $b, c \in \Sigma$, whether or not

$$(\forall w \in L(A)) \ \mid |w|_b - |w|_c \mid \leq p(|w|). \tag{8}$$

In the following let $b, c \in \Sigma$ be fixed. By our assumptions concerning the polynomial $p(x)$ we can effectively find $M \in \mathbb{N}$ such that

$$p(x) > x \ \text{ for all } x > M.$$

Since $\mid |w|_b - |w|_c \mid$ cannot be greater than $|w|$, in order to decide (8), it is sufficient to test the condition for words of $L(A)$ of length at most M. □

Intuitively, we can say that a super-linear fairness condition is satisfied unless it is violated by some word of constant length, where the constant depends only on the fairness function. It can be noted that the decision algorithm given by the above proof is extremely inefficient since it uses an exhaustive search over all words of at most a certain length.

Finally, we may note that [11] considered also a notion called *initial fairness*, where the fairness condition for symbols $b, c \in \Sigma$ is required to hold only "as long as" the remaining suffix contains occurrences of both symbols b and c. It was shown that the uniform initial contant-fairness question is decidable for context-free languages. The proof of Theorem 3.2 could fairly easily be modified to show that also the initial linear fairness (or polynomial fairness) condition is decidable. However, we feel that the notion of initial fairness is well motivated, perhaps, only in the constant case. The reason is that the standard constant fairness condition by itself is very restrictive as it requires that all words contain almost the same number of occurrences of arbitrary symbols b and c.

References

[1] J.-M. Autebert, J. Berstel and L. Boasson, Context-free languages and push-down automata, in: *Handbook of Formal Languages, Vol. I.* (G. Rozenberg, A. Salomaa, eds.), pp. 111–174, Springer-Verlag, 1997.

[2] H.-D. Burkhard, Fairness and control in multi-agent systems, *Theoret. Comput. Sci.* **189** (1997) 109–127.

[3] N. Francez, *Fairness,* Springer-Verlag, Berlin, 1986.

[4] M.A. Harrison, *Introduction to formal language theory,* Addison-Wesley, Reading, MA, 1978.

[5] C. Hartonas, A fixpoint approach to finite delay and fairness, *Theoret. Comput. Sci.* **198** (1998) 131–158.

[6] M. Kudlek and A. Mateescu, On distributed catenation, *Theoret. Comput. Sci.* **180** (1997) 341–352.

[7] W. Kuich, Semirings and formal power series, in: *Handbook of Formal Languages, Vol. I.* (G. Rozenberg, A. Salomaa, eds.), pp. 609–677, Springer-Verlag, 1997.

[8] A. Mateescu, CD grammar systems and trajectories, *Acta Cybernetica* **13** (1997) 141–157.

[9] A. Mateescu, G.D. Mateescu, G. Rozenberg and A. Salomaa, Shuffle-like operations on ω-words, manuscript 1996.

[10] A. Mateescu, G. Rozenberg and A. Salomaa, Shuffle on trajectories: Syntactic constraints, *Theoret. Comput. Sci.* **197** (1998) 1–56.

[11] A. Mateescu, K. Salomaa and S. Yu, Decidability of fairness for context-free languages, in: *Proceedings of the Third International Conference on Developments in Language Theory, DLT'97 (Thessaloniki, July 20–23, 1997),* S. Bozapalidis (ed.), pp. 351–364.

[12] R.J. Parikh, On context-free languages, *J. Assoc. Comput. Mach.* **13** (1966) 570–581.

[13] D. Park, Concurrency and automata on infinite sequences, in: "Theoretical Computer Science", *Proc. of the 5th GI Conference,* P. Deussen (ed.), Lect. Notes Comput. Sci. **104**, Springer-Verlag, (1981) 167–183.

[14] L. Priese, R. Rehrmann and U. Willecke-Klemme, An introduction to the regular theory of fairness, *Theoret. Comput. Sci.* **54** (1987) 139–163.

[15] J. Romijn and F. Vaandrager, A note on fairness in I/O automata, *Inform. Process. Lett.* **59** (1996) 245–250.

[16] A. Salomaa, *Formal languages,* Academic Press, New York, 1973.

[17] S. Yu, Regular languages, in: *Handbook of Formal Languages, Vol. I.* (G. Rozenberg, A. Salomaa, eds.), pp. 41–110, Springer-Verlag, 1997.