ALTERNATING REGULAR TREE GRAMMARS IN THE FRAMEWORK OF LATTICE-VALUED LOGIC

M. GHORANI AND M. M. ZAHEDI

Abstract. In this paper, two different ways of introducing alternation for lattice-valued (referred to as $\mathscr{L}$-valued) regular tree grammars and $\mathscr{L}$-valued top-down tree automata are compared. One is the way which defines the alternating regular tree grammar, i.e., alternation is governed by the non-terminals of the grammar and the other is the way which combines state with alternation. The first way is taken over to prove a main theorem: the class of languages generated by an $\mathscr{L}$-valued alternating regular tree grammar ($\mathcal{LAG}$) is equal to the class of languages accepted by an $\mathscr{L}$-valued alternating top-down tree automaton ($\mathcal{LAA}$). The second way is taken over to define a new type of automaton by combining the $\mathscr{L}$-valued alternating top-down tree automaton with stack, called $\mathscr{L}$-valued alternating stack tree automaton ($\mathcal{LASA}$) and the generative power of it is compared to some well-known language classes, especially to $\mathcal{LAA}$ and to $\mathcal{LAG}$. Also, we have derived a characterization of the state alternating regular tree grammar ($\mathcal{LSAG}$) in terms of $\mathcal{LASA}$.

1. Introduction

Tree automata came out of the algebraic theory of finite automata by Büchi [4] in the 1960's. Tree automata techniques have been commonly used in checking consistency of tree structures. Typical examples include checking sufficient completeness of algebraic specifications [1] and the consistency of semi-structured documents [15]. Syntactic pattern recognition, logic programming, term rewriting and linguistics are some other areas in which tree automata have been used [8, 11].

The mathematical formulation of a fuzzy automaton was introduced by Wee [48] and Santos [39]. Subsequently, the fundamentals of fuzzy language theory were established by Lee and Zadeh [21], and by Thomason and Marinos [43]. Thereafter, many other authors having contributed to this field (see [16, 24, 27, 36, 42]). Fuzzy automata have many significant applications in different subject such as learning systems [33], fuzzy discrete event systems [26, 37] and neural networks [27, 32]. Due to the importance of residuated lattice logic, Qiu [34, 35] established a fundamental framework of automata theory based on complete residuated lattice-valued logic. He and others [38, 50, 51, 52, 53] have studied equivalence problem, reduction of states, context-free grammars, and pushdown automata in this framework. In recent years, several authors have studied lattice-valued automata [17, 22, 23, 25].

Received: August 2014; Revised: August 2015; Accepted: January 2016

Key words and phrases: Lattice-valued logic, Alternating top-down tree automaton, State alternating regular tree grammar, Alternating stack tree automaton.

The concept of fuzzy tree automata considered by many authors [3, 5, 7, 9]. Algebraic study of fuzzy tree automata was done in [3, 9]. Since, fuzzy tree automata take values in the unit interval [0, 1], to enhance the processing ability of fuzzy tree automata, Ghorani and Zahedi [12] and Ghorani et al. [13] extended the membership value to a more general algebraic structure and considered tree automata based on complete residuated lattice-valued logic.

Alternation is a powerful generalization of nondeterminism with many applications in automata and complexity theory. It has been considered for long time as a computation model, e.g. for Turing machines. The seminal work in this area was done by Chandra et al. [6]. They studied the relationship between complexity classes defined by (non-)deterministic automata and alternating automata. Slutzki [40, 41] investigated the effect of alternation on several kinds of tree automata. Alternating automata are efficient data structures for many problems in the specification and verification of reactive systems. Because of their succinctness they are a convenient way to represent temporal specifications [45, 46]. They also have been commonly used as a basis for static verification [44]. In [10] it has been shown that how alternating automata can be used in runtime verification. Moreover, the connection of alternating tree automata, parity games and μ-calculus considered in [18, 49].

The notion of alternating pushdown automata was investigated by Ladner et al. [20, 19]. Moriya [28] introduced the concept of alternating context free grammar and showed that the class of alternating context free languages is equal to the class of languages accepted by alternating pushdown automata. In [29], the authors defined the concepts of state alternating context free grammar and extended alternating context free grammar, and proved that these grammars are equivalent. Moriya and Otto [30] introduced the concept of stack-alternating pushdown automata and showed that these automata have the same accepting power as the ordinary alternating pushdown automata. Moriya and Otto [31] extended the notion of context free grammar to phrase-structure grammar and obtained some results for phrase-structure grammars. Also, Verma and Goubault-Larrecq [47] considered the notion of alternating two-way AC-tree automata. Some basic properties of alternating tree automata explained in [8].

Now in this paper we follow [12, 13, 29, 30]. At first the concepts of $\mathcal{LAG}$ and $\mathscr{L}$-valued alternating normalized regular tree grammar ($\mathcal{LANG}$) are defined and the equivalence between them is shown. Furthermore, it is shown that the language generated by an $\mathcal{LAG}$ coincides with the language accepted by an $\mathcal{LAA}$. However, there are languages that can not be accepted by an $\mathcal{LAA}$. Thus, the concept of $\mathcal{LSAG}$ is defined and is compared to $\mathcal{LAG}$. In particular, the basic fact that $\mathcal{LSAG}s$ are more powerful than $\mathcal{LAG}s$ in their generative capacity is obtained. In the definition of an $\mathcal{LAG}$, alternation is governed by the non-terminals of the grammar, while alternation is ruled by states in an $\mathcal{LSAG}$. Also, we introduce an $\mathscr{L}$-valued alternating stack tree automaton and claim that a language is acceptable by this automaton if and only if it can be generated by an $\mathcal{LSAG}$. Since an automaton is a special tree automaton, our results are the extension of the related results in automata to tree automata.

The rest of this paper is arranged as follows: Section 2 contains some preliminaries and basic definitions. In Section 3 we will compare the $\mathcal{LAG}s$ to the $\mathcal{LAA}s$, and will obtained a main theorem: an $\mathscr{L}$-valued alternating tree language is acceptable if and only if it is regular. In Section 4, $\mathscr{L}$-valued state alternating regular tree grammar is considered. In Section 5, basic properties of $\mathcal{LAG}s$ and $\mathcal{LSAG}s$ are studied and an interesting theorem is obtained: the class of $\mathcal{LSAG}$ languages is precisely the class of $\mathcal{LASA}$ languages. Finally, in Section 6 conclusions are given.

2. **Preliminaries and Basic Definitions**

We assume that the readers are familiar with the basic concepts in tree automata and regular tree grammars. For more details we refer to [8, 11, 12, 13, 14, 28, 38, 41].

Definition 2.1. [2] A complete residuated lattice is a 5-tuple $l =< \mathscr{L}, +, \cdot, \odot, \rho >$ where:

(i) $< \mathscr{L}, +, \cdot >$ is a complete lattice with the least and greatest elements 0 and 1, respectively,

(ii) $\odot$ and ρ are two binary operations on $\mathscr{L}$ such that $\odot$ is isotone and $< \mathscr{L}, \odot, 1 >$ is a commutative monoid, and ρ is antitone in the first and isotone in the second variable, that is, for any $a_1, a_2, b \in \mathscr{L}$ if $a_1 \leq a_2$ then $a_1 \odot b \leq a_2 \odot b$, $b \odot a_1 \leq b \odot a_2$, $a_2 \rho b \leq a_1 \rho b$ and $b \rho a_1 \leq b \rho a_2$,

(iii) for all $a, b, c \in \mathscr{L}$, $a \odot b \leq c$ if and only if $a \leq b \rho c$.

Example 2.2. Let $\mathscr{L} = [0, 1] \subseteq \mathbb{R}$. Then $l =< \mathscr{L}, \vee, \wedge, \odot, \rho >$ is a complete residuated lattice, where $a \odot b = \max(0, a + b - 1), a \rho b = \min(1, 1 - a + b)$ for any $a, b \in [0, 1]$, $\vee$ and $\wedge$ are the symbols of the truth-valued lattice, representing max and min, respectively.

Remark 2.3. In this paper, by an $\mathscr{L}$-valued logic we mean a complete residuated lattice-valued logic; that is, the set of truth values is $\mathscr{L}$, which possesses nullary connective a ($a \in \mathscr{L}$), an additional binary connective & as well as usual connectives $\vee, \wedge$ and implication $\rightarrow$.

Definition 2.4. [8] A ranked alphabet is a couple $(F, Arity)$ where F is a finite set and $Arity$ is a mapping from F into $\mathbb{N}$ (the set of positive integers). The arity of a symbol $f \in F$ is $Arity(f)$. The set of symbols of arity n is denoted by F_n. Elements of arity $0, 1, \cdots, n$ are respectively called constants, unary,$\cdots$, $n-$ary symbols.

We assume that F contains at least one constant. Here, we use parenthesis and commas for a short declaration of symbols with arity. For instance, $f(,)$ is a short declaration for a binary symbol f.

Definition 2.5. [8] The set of trees over F, denoted by $T(F)$, is the smallest set defined as follows:
(1) $F_0 \subseteq T(F)$,
(2) if $n \geq 1, \sigma \in F_n$, and $t_1, \cdots, t_n \in T(F)$, then $\sigma(t_1, \cdots, t_n) \in T(F)$.

Definition 2.6. [41] Assume that H is a set of symbols or trees. The set of trees over F indexed by H, denoted by $T(F)[H]$, is the smallest set defined as follows:

(1) $H \cup F_0 \subseteq T(F)[H]$,
(2) if $n \geq 1$, $\sigma \in F_n$ and $t_1, \cdots, t_n \in T(F)[H]$, then $\sigma(t_1, \cdots, t_n) \in T(F)[H]$.

Definition 2.7. [8] Let X_n be a set of n variables. A linear tree $c \in T(F, X_n)$ is called a context and the expression $c[t_1, \cdots, t_n]$ for $t_1, \cdots, t_n \in T(F)$ denotes the tree in $T(F)$ obtained from c by replacing variable x_i by t_i for each $1 \leq i \leq n$, that is $c[t_1, \cdots, t_n] = c\{x_1 \leftarrow t_1, \cdots, x_n \leftarrow t_n\}$. We denote by $C^n(F)$ the set of contexts over $(x_1, \cdots, x_n)$, and by $C(F)$ the set of contexts containing a single variable.

3. $\mathscr{L}$-valued Alternating Regular Tree Grammar

Slutzki [40, 41] defined alternating top-down tree automaton. Now, we define an $\mathscr{L}$-valued alternating top-down tree automaton. Note that from now on we assume that $c \in \mathscr{L}$.

Definition 3.1. An $\mathscr{L}$-valued alternating top-down tree automaton ($\mathcal{LAA}$) is a tuple $\mathcal{A} = (Q, U, F, q_0, \delta)$, where Q is a finite non-empty set of states, $U \subseteq Q$ is a set of universal states (states in $Q \setminus U$ are called existential states), F is a ranked alphabet, $q_0 \in Q$ is the initial state, and δ is a set of $\mathscr{L}$-valued rules of the form:

$$q(f(x_1, \cdots, x_n)) \rightarrow^c f(q_1(x_1), \cdots, q_n(x_n)),$$

where $n \geq 0$, $f \in F_n$, $q, q_1, \cdots, q_n \in Q$, $x_1, , \cdots, x_n \in X$ and X is the set of variables.

Definition 3.2. An instantaneous description (ID) of $\mathcal{A}$ on a tree $t \in T(F)$ is a tree in the set $T(F)[Q(T(F))]$, where $Q(T(F))$ is the set $\{q(t) |\ q \in Q,\ t \in T(F)\}$, $q_0(t)$ is the initial ID of $\mathcal{A}$ on t, and trees in $T(F)$ are the accepting IDs. For two IDs s and r we write $s \rightarrow r$ if there exists a rule $(q(f(x_1, \cdots, x_n)) \rightarrow^c f(q_1(x_1), \cdots, q_n(x_n))) \in \delta$ such that r is obtained from s by replacing a subtree of s of the form $q(f(t_1, \cdots, t_n))$, for certain $t_1, \cdots, t_n \in T(F)$, with $f(q_1(t_1), \cdots, q_n(t_n))$.

Definition 3.3. Let $\mathbb{N}^*$ be the set of finite strings over $\mathbb{N}$, and ε be the empty string. A computation tree $\mathcal{T} \in T(T(F)[Q(T(F))])$ is defined as a partial function $\mathcal{T} : \mathbb{N}^* \rightarrow T(F)[Q(T(F))]$ with domain $pos(\mathcal{T})$ satisfying the following properties:

(a) $t(\varepsilon) = Head(t)$, where $Head(t)$ is the root symbol of $\mathcal{T}$,
(b) $pos(\mathcal{T})$ is non-empty and prefix-closed,
(c) $\forall p \in pos(\mathcal{T})$, if $\mathcal{T}(p) \in T(F)[Q(T(F)] \setminus T(F)$, then $\{j | pj \in pos(\mathcal{T})\} = \{1, \cdots, n\}$, where n is the number of sons of $\mathcal{T}(p)$,
(d) $\forall p \in pos(\mathcal{T})$, if $\mathcal{T}(p) \in T(F)$, then $\{j | pj \in pos(\mathcal{T})\} = \emptyset$.

Each element in $pos(\mathcal{T})$ is called a position.

Remark 3.4. Note that if $r_1, \cdots, r_m$ be all the rules in δ that have $q(f(x_1, \cdots, x_n))$ as a left-hand side such that $q(f(x_1, \cdots, x_n))$ is a subtree in $\mathcal{T}(p)$ then:

- if q is a universal state, then by applying all the rules $r_i,\ i = 1, \cdots, m$, $\mathcal{T}(p)$ will have m sons $s_i,\ i = 1, \cdots, m$,
- if q is an existential state, then by applying one of the rules $r_i,\ i = 1, \cdots, m$, $\mathcal{T}(p)$ will have a single son.

Definition 3.5. An accepting computation tree of $\mathcal{A}$ on t is a computation tree of $\mathcal{A}$ on t whose leaves are all labelled by accepting IDs and $Head(t) = q_0(t)$. $\mathcal{A}$ accepts t, if there exists an accepting computation tree of $\mathcal{A}$ on t. The tree language defined by $\mathcal{A}$ is

$$L(\mathcal{A}) = \{(t, \mu_A(t))|\ \mathcal{A} \text{ accepts } t \in T(F)\},$$

where, $\mu_A(t)$ (the membership value of t) is obtained as follows:
let the computation tree $\mathcal{T}$ contain k paths $p_1, \cdots, p_k$. Also, let each path $p_j, j = 1, \cdots, k$ contain $r_j, j = 1, \cdots, k$ positions, named $s_1^j, \cdots, s_{r_j}^j$, where $s_i^j < s_{i+1}^j,\ i = 1, \cdots, r_j - 1$. The membership value of each path $p_j,\ j = 1, \cdots, k$ is calculated as $\mu(p_j) = \wedge_{i=1}^{r_j-1} c_i^j$, where $\mathcal{T}(s_i^j) \rightarrow^{c_i^j} \mathcal{T}(s_{i+1}^j)$. Finally, $\mu_A(t) = \vee \vee_j \mu(p_j)$ and supremum is taken over all computation trees from $q_0(t)$ to t.

Problem. Can we derive a new grammatical characterization for the class of languages that are accepted by an $\mathcal{LAA}$? The answer is positive. In this section, we answer to this question.

Definition 3.6. An $\mathscr{L}$-valued alternating regular tree grammar ($\mathcal{LAG}$) is a 5-tuple $G = (S, N, U, E, R)$, where S is the start symbol, N is a set of non-terminal symbols, $U \subseteq N$ is a subset of non-terminals, E is a set of terminal symbols, and R is a set of $\mathscr{L}$-valued productions of the form $A \rightarrow^c \beta$, where A is a non-terminal of N and β is a tree of $T(E \cup N)$. The non-terminals in U are called universal non-terminals and non-terminals in $N \setminus U$ are called existential non-terminals.

In the above definition if all of the non-terminals in $\mathcal{LAG}$ are existential, we receive to $\mathscr{L}$-valued regular tree grammar, for short $\mathcal{LG}$. Let G be an $\mathcal{LAG}$ and $\alpha \in T(E \cup N)$. A finite tree $\mathcal{T}$ is a derivation tree for G from α if the following properties are satisfied:
(1) each node *positiveimplicative* of $\mathcal{T}$ is labelled with a tree in $T(E \cup N)$, the root of $\mathcal{T}$ is labelled with α,
(2) if *positiveimplicative* is an internal node of $\mathcal{T}$ with label $f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n)$, where $f \in E, u_j \in T(E \cup N), j = 1, \cdots, i-1, i+1, \cdots, n$, w is an existential non-terminal and $w \rightarrow^c \alpha$ is a production from R, then

$$f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n) \Rightarrow^c f(u_1, \cdots, u_{i-1}, \alpha, u_{i+1}, \cdots, u_n),$$

(3) if w is a universal non-terminal and $w \rightarrow^{c_j} \alpha_j,\ j = 1, \cdots, k$ are all the productions with left-hand side w, then

$$f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n) \Rightarrow^{(c_1, \cdots, c_k)} (f(u_1, \cdots, u_{i-1}, \alpha_1, u_{i+1}, \cdots, u_n),$$
$$\cdots, f(u_1, \cdots, u_{i-1}, \alpha_k, u_{i+1}, \cdots, u_n)),$$

that is, all productions with left-hand side w are applied simultaneously. Note that, $f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n)$ contains k sons and $c_i,\ i = 1, \cdots, k$ is the membership value of the ith son.
It is important to mention that a derivation tree $\mathcal{T}$ can also be defined as follows: $\mathcal{T} \in T(T(E \cup N))$ is defined as a partial function $\mathcal{T} : \mathbb{N}^* \rightarrow T(E \cup N)$ with domain $pos(\mathcal{T})$ defined similar to Definition 3.3.

If there exists a derivation tree such that the root is labelled with the start symbol S and all leaves are labelled with the tree t, then we say that the terminal tree t derives from G. The language generated by $\mathcal{LAG}$ G, denoted by $L(G)$, is the set of all terminal trees derived from G. The membership value of t in $L(G)$ is obtained as follows:
Let the derivation tree $\mathcal{T}$ contain k paths $p_1, \cdots, p_k$. Also, assume that each path $p_j, j = 1, \cdots, k$ contains $r_j, j = 1, \cdots, k$ positions, named $s_i^j,\ i = 1, \cdots, r_j$, where $s_i^j < s_{i+1}^j$. The membership value of each path $p_j,\ j = 1, \cdots, k$ is calculated as $\mu(p_j) = \wedge_{i=1}^{r_j-1} c_i^j$, where $\mathcal{T}(s_i^j) \rightarrow^{c_i^j} \mathcal{T}(s_{i+1}^j)$. Finally, $\mu_G(t) = \vee \vee_j \mu(p_j)$ and the supremum is taken over all derivation trees from S to t. Therefore

$$L(G) = \{(t, \mu_G(t))|\ t \text{ derives from } G\}.$$

Definition 3.7. An acceptable $\mathscr{L}$-valued alternating tree language is a language accepted by an $\mathcal{LAA}$. Also, an $\mathscr{L}$-valued alternating regular tree language is a language generated by an $\mathcal{LAG}$.

Definition 3.8. An $\mathcal{LAG}$ G is said to be linear if the right-hand side of each production contains at most one non-terminal occurrence.

Definition 3.9. Two $\mathcal{LAG}$s G_1 and G_2 are equivalent if they generate the same language.

Definition 3.10. An $\mathscr{L}$-valued alternating normalized regular tree grammar ($\mathcal{LANG}$) $G = (S, N, U, E, R)$ is an $\mathcal{LAG}$ in which the productions are in the form $A \rightarrow^c a$ or $A \rightarrow^c f(A_1, \ldots, A_n)$, where f and a are symbols of E and $A_1, \cdots, A_n, A$ are non-terminals.

Theorem 3.11. *Each $\mathcal{LAG}$ G is equivalent to an $\mathcal{LANG}$ G'.*

Proof. Let $A \rightarrow^c f(s_1, \cdots, s_m)$ be a production in G. We replace this production with $A \rightarrow^c f(A_1, \cdots, A_m)$ in G', such that if $s_j \in N$ then $A_j = s_j$, otherwise, A_j is a new non-terminal symbol and we add the production $A_j \rightarrow^1 s_j$. We iterate this process until we reach to a grammar with productions of the form $A \rightarrow^c f(A_1, \cdots, A_m)$, $A \rightarrow^c a$ or $A \rightarrow^c B$. Finally, we replace the productions $A_1 \rightarrow^{c_1} A_2,\ A_2 \rightarrow^{c_2} A_3, \cdots, A_k \rightarrow^{c_k} a$ with the production $A_1 \rightarrow^c a$, such that $c = \wedge_{j=1}^{k} c_j$, for all $a \notin N$, where $A_j \in Q \setminus U$. Meanwhile, if A_j is a universal non-terminal symbol, for $1 \leq j \leq k$, then we consider A_1 as a universal symbol. Also, for other cases we can reach to a grammar with productions of the form $A \rightarrow^c f(A_1, \cdots, A_m)$ or $A \rightarrow^c a$. Thus, $L(G) \subseteq L(G')$. It is obvious that $L(G') \subseteq L(G)$. Hence, $L(G) = L(G')$. □

By Theorem 3.11, we can consider the same properties for $\mathcal{LAG}$ and $\mathcal{LANG}$.

The following lemma shows that there exists an $\mathcal{LAA}$, which accepts the language generated by a given $\mathcal{LAG}$.

Lemma 3.12. *For each $\mathscr{L}$-valued alternating regular tree language $\bar{L}$, there exists an $\mathcal{LAA}$ $\mathcal{A}$ such that $\bar{L} = L(\mathcal{A})$.*

Proof. Let $G = (S, N, U, E, R)$ be an $\mathcal{LANG}$ with $\bar{L} = L(G)$. We construct an $\mathcal{LAA}$ $\mathcal{A} = (Q, U', F, q_S, \delta)$ as follows:

- $Q = \{q_A | A \in N\}$,
- $F = E$,
- $U' = \{q_A | A \in U\}$, and
- δ consists of the following rules:

  $A \to^c f(A_1, \cdots, A_n) \in R$ if and only if

  $$q_A(f(x_1, \cdots, x_n)) \to^c f(q_{A_1}(x_1), \cdots, q_{A_n}(x_n)) \in \delta.$$

  If $n = 0$, then

  $$(A \to^c a) \in R \text{ if and only if } (q_A(a) \to^c a) \in \delta.$$

Now, we complete the proof by induction on the length of derivation. Let $(t, c) \in L(G)$. Then, there exists a derivation tree $\mathcal{T}$ of G such that its root is labelled with S and each leaf is labelled with t. Let the length of derivation tree $\mathcal{T}$ be n.

Base case. Suppose $n = 1$, therefore, $t = a \in F_0$. Thus there exists a computation tree $\mathcal{T}'$ of G' from $q_S(a)$ to a.

Induction step. Suppose that the property holds for any derivation of length less than n and let $t = f(t_1, \cdots, t_k)$. The first step of derivation from S is as follows:

$$S \Rightarrow (s_1, \cdots, s_m).$$

It is obvious that $s_i,$ with $1 \leq i \leq m$, must be as follows:

$$s_1 = f(A_{11}, \cdots, A_{1k}),$$

$$\vdots$$

$$s_m = f(A_{m1}, \cdots, A_{mk}).$$

Therefore there exist some derivation trees from A_{rj} to t_j in n_{rj} steps. By induction hypothesis, we conclude that there exist computation trees from $q_{A_{rj}}(t_j)$ to t_j in n_{rj} steps where $1 \leq r \leq m,\ 1 \leq j \leq k$, and $\sum_{r=1}^{m} \sum_{j=1}^{k} n_{rj} = n - 1$. Therefore, there exist computation trees from $f(q_{A_{r1}}(t_1), \cdots, q_{A_{rk}}(t_k))$ to $f(t_1, \cdots, t_k)$ with derivation length $\sum_{j=1}^{k} n_{rj}$, $\forall r = 1, \cdots, m$. Hence, there exists an accepting computation tree $\mathcal{T}'$ of G' such that its root is labelled with $q_S(t)$ and each leaf is labelled with t with membership value c. Thus, $L(G) \subseteq L(\mathcal{A})$. By a similar method, it is easy to show that $L(\mathcal{A}) \subseteq L(G)$. Consequently, $L(G) = L(\mathcal{A})$. □

The following lemma shows that there exists an $\mathcal{LAG}$, which generates the language accepted by a given $\mathcal{LAA}$.

Lemma 3.13. *For each $\mathcal{LAA}$ $\mathcal{A}$, there exists an $\mathcal{LAG}$ G such that $L(\mathcal{A}) = L(G)$.*

Proof. Let $\mathcal{A} = (Q, U, F, q_0, \delta)$ be an $\mathcal{LAA}$. We define $G = (S, N, U', E, R)$ as follows:

- S is a new symbol,
- $N = \{A_q |\ q \in Q\}$,
- $E = F$,
- $U' = \{A_q |\ q \in U\}$, and

- $R = \{A_q \to^c f(A_{q_1}, \cdots, A_{q_n}) |\ q(f(x_1, \cdots, x_n)) \to^c f(q_1(x_1), \cdots, q_n(x_n)) \in \delta\} \cup \{S \to^1 A_{q_0}\}$.

The proof can be completed by induction on the length of derivation, similar to the proof of Lemma 3.12. □

Now, combining Lemmas 3.12 and 3.13, we get the equivalence between acceptability and regularity of $\mathscr{L}$-valued alternating tree languages.

Theorem 3.14. *An $\mathscr{L}$-valued alternating tree language is acceptable if and only if it is regular.*

Remark 3.15. In [8], the equivalence between acceptability and regularity of tree languages has been shown. Theorem 3.14 extends that result to $\mathscr{L}$-valued alternating tree languages.

In the following we give an example to clarify the above theorem.

Example 3.16. Let $\mathscr{L} = [0, 1]$ and $G = (S, N, U, E, R)$ be an $\mathcal{LANG}$, where

$$N = \{S, A, B\},\ U = \{S\},\ E = \{f(), a\},$$

and R consists of the following production rules:

$$S \to^{0.2} f(A),\ S \to^{0.3} f(B),\ A \to^{0.2} f(A),\ A \to^{0.4} a,\ B \to^{0.3} a.$$

Then any derivation tree in G must be of the form shown in Figure 1, where m is a positive integer. By using the derivation given in Figure 1 it is obvious that, the only terminal tree derived from G is $f(a)$. Thus $L(G) = \{(f(a), 0.3)\}$.

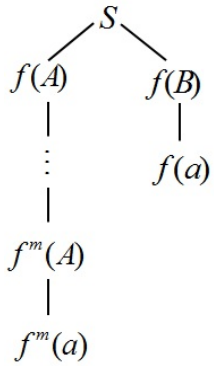

FIGURE 1. A Derivation Tree of Example 3.16

Now, we construct the related $\mathcal{LAA}$ $\mathcal{A} = (Q, U', E, q_S, \delta)$ where

$$Q = \{q_S, q_A, q_B\},\ U' = \{q_S\},$$

and the production rules are as the following:

$$q_S(f(x_1)) \to^{0.2} f(q_A(x_1)),$$

$$q_S(f(x_1)) \to^{0.3} f(q_B(x_1)),$$
$$q_A(f(x_1)) \to^{0.2} f(q_A(x_1)),$$
$$q_A(a) \to^{0.4} a,$$
$$q_B(a) \to^{0.3} a.$$

Thus, we have only one accepting computation tree of $\mathcal{A}$ which is on $f(a)$ as Figure 2. Therefore, $L(\mathcal{A}) = \{(f(a), 0.3)\}$. Hence $L(G) = L(\mathcal{A})$.

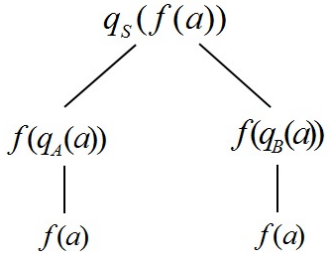

FIGURE 2. A Computation Tree of Example 3.16

Conversely, let $\mathcal{A} = (Q, U, E, q_0, \delta)$ be an $\mathscr{L}$-valued alternating top-down tree automaton, where

$$Q = \{q_0, q_1, q_2\},\ U = \{q_0\},\ E = \{f(), a\},$$

and δ consists of the following rules:

$$q_o(f(x_1)) \to^{0.2} f(q_1(x_1)),$$
$$q_0(f(x_1)) \to^{0.4} f(q_2(x_1)),$$
$$q_1(f(x_1)) \to^{0.6} f(q_1(x_1)),$$
$$q_1(a) \to^{0.1} a,$$
$$q_2(a) \to^{0.2} a.$$

Then, the accepting computation tree of $\mathcal{A}$ on $f(a)$ is of the form given in Figure 3. Clearly $L(\mathcal{A}) = \{(f(a), 0.2)\}$.

Now, we construct the $\mathcal{LAFG}$ $G = (S, N, U', E, R)$ where

$$N = \{A_{q_0}, A_{q_1}, A_{q_2}\},\ U' = \{A_{q_0}\},$$

and R consists of the following production rules:

$$S \to^1 A_{q_0},\ A_{q_0} \to^{0.2} f(A_{q_1}),\ A_{q_0} \to^{0.4} f(A_{q_2}),$$
$$A_{q_1} \to^{0.6} f(A_{q_1}),\ A_{q_1} \to^{0.1} a,\ A_{q_2} \to^{0.2} a.$$

Then, any derivation tree in G must be of the form shown in Figure 4, where m is a positive integer.

The only terminal tree derived from G is $f(a)$. Thus, $L(G) = \{(f(a), 0.2)\}$. Therefore, $L(G) = L(\mathcal{A})$.

We now give an example of a tree language which is not acceptable by an $\mathcal{LAA}$.

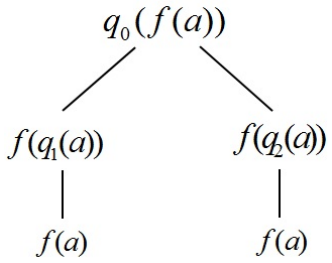

FIGURE 3. A Computation Tree of $\mathcal{A}$ of Example 3.16

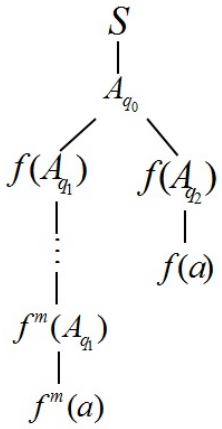

FIGURE 4. A Derivation Tree of Example 3.16

Example 3.17. Let $\mathscr{L} = [0,1]$ and $F = \{f(,), g(), a\}$. Consider the following $\mathscr{L}$-valued alternating tree language:

$$\bar{L} = \{(f(g^n(a), g^n(a)), 0.3) | n \geq 0\}.$$

Assume that $\bar{L}$ is acceptable by an $\mathcal{LAA}$ $\mathcal{A}$. Now, consider the tree $t = f(g^i(a), g^i(a)) \in \bar{L}$ with $|Q| \leq i$. Therefore, there exists a computation tree from $q(g^i(a))$ to $g^i(a)$, where $q \in Q$. Hence, during the mentioned computation tree, after some steps we reach from $q(g^i(a))$ to $g^m(q(g^k(a)))$ and then to $g^i(a)$, where $m + k = i, m \neq 0$. Thus, there exists a computation tree from $q(g^k(a))$ to $g^k(a)$. Therefore $f(g^k(a), g^i(a)), k < i$ is acceptable by $\mathcal{A}$, which is a contradiction.

The above example, shows that there are languages which can not be generated by $\mathcal{LAG}s$. Therefore, we have to seek grammars beyond these grammars. Hence, in the next section we will find grammars that are more powerful than $\mathcal{LAG}s$.

4. $\mathscr{L}$-valued State Alternating Regular Tree Grammar

An $\mathscr{L}$-valued extended regular tree grammar ($\mathcal{LEG}$) is a 6-tuple $G = (Q, N, E, S, q_0, R)$, where Q is a finite set of states, N is a finite set of non-terminal symbols, E is a finite set of terminal symbols, $S \in N$ is the start symbol, $q_0 \in Q$ is the initial state, and R is a finite set of productions of the form $(p, w) \rightarrow^c (q, \alpha)$, where $p, q \in Q$, $w \in N$, $c \in L$ and $\alpha \in T(E \cup N)$. In fact an $\mathcal{LEG}$ is a combination of $\mathscr{L}$-valued regular tree grammar with states. A derivation tree $\mathcal{T} \in T(E \cup N)$ is defined as a partial function $\mathcal{T} : \mathbb{N}^* \rightarrow T(E \cup N)$ with domain $pos(\mathcal{T})$ defined similar to Definition 3.3.

In the following definition, we combine the notion of alternation with an $\mathscr{L}$-valued regular tree grammar with states.

Definition 4.1. An $\mathscr{L}$-valued extended alternating regular tree grammar ($\mathcal{LEAG}$) is a tuple $G = (Q, S, N, U, E, R, q_0, Q_f)$, where S, N and E are defined as in Definition 3.6, Q is a finite non-empty set of states, $q_0 \in Q$ is the initial state, $U \subseteq N$ is a set of universal non-terminals, $Q_f \subseteq Q$ is a set of final states, and R is a finite set of productions of the form $(p, w) \rightarrow^c (q, \alpha)$, where $p, q \in Q$, $w \in N$ and $\alpha \in T(E \cup N)$.

A derivation tree $\mathcal{T} \in T(Q, T(E \cup N))$ is defined as a partial function $\mathcal{T} : \mathbb{N}^* \rightarrow (Q, T(E \cup N))$ with domain $pos(\mathcal{T})$ defined similar to Definition 3.3.

Remark 4.2. Note that if $(p, w) \rightarrow^{c_j} (q_j, \alpha_j)$, where $w \in N$, $p, q_j \in Q$ and $j = 1, \cdots, k$, are all the productions in R that have (p, w) as a left-hand side such that $(p, f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n))$ is a tree in $\mathcal{T}(p)$, then:

- if w is a universal non-terminal, then by applying all of the productions $(p, w) \rightarrow^{c_j} (q_j, \alpha_j)$, $j = 1, \cdots, k$, $\mathcal{T}(p)$ will have k sons

$$((q_1, f(u_1, \cdots, u_{i-1}, \alpha_1, u_{i+1}, \cdots, u_n)), \cdots, (q_k, f(u_1, \cdots, u_{i-1}, \alpha_k, u_{i+1}, \cdots, u_n))),$$

  with membership value $(c_1, \cdots, c_k)$,
- if w is an existential non-terminal, then by applying one of the productions $(p, w) \rightarrow^{c_j} (q_j, \alpha_j)$, $j = 1, \cdots, k$, $\mathcal{T}(p)$ will have a single son.

If there exists a derivation tree such that the root is labelled with the start symbol (q_0, S) and all leaves are labelled with pairs of the form (p, t) with $p \in Q_f$, then the terminal tree t can be derived from G. An $\mathcal{LEAG}$ G generates a language $L(G)$ in the following manner:
a terminal tree t is in $L(G)$ if and only if it is derived from (q_0, S). The membership value of t in $L(G)$ is calculated by a method similar to that of a language generated by an $\mathcal{LAG}$.

Note that the labels of different leaves may differ in their first components, but they must agree in their second components.

$\mathcal{LEAG}s$ are obtained from $\mathcal{LAG}s$ by introducing states, in the same way that $\mathcal{LEG}s$ are obtained from $\mathscr{L}$-valued regular tree grammars. Also $\mathcal{LEAG}s$ are obtained from $\mathcal{LEG}s$ by distinguishing between universal and existential non-terminals, in the same way that $\mathcal{LAG}$ are obtained from $\mathscr{L}$-valued regular tree grammars. Hence, the $\mathcal{LEAG}s$ unify these two generalizations of $\mathcal{LAG}s$.

As it will be seen, the concept of $\mathcal{LEAG}$ is equivalent to the following, where alternation is governed by states and not by non-terminals.

Definition 4.3. An $\mathscr{L}$-valued state alternating regular tree grammar ($\mathcal{LSAG}$) is an 8-tuple $G = (Q, U, S, N, E, R, q_0, Q_f)$, where Q is a finite non-empty set of states, $U \subseteq Q$ is a set of universal states, N is a finite set of non-terminal symbols, E is a finite set of terminal symbols, $S \in N$ is the start symbol, R is a set of productions of the form $(p, w) \to^c (q, \alpha)$, where $p, q \in Q$, $w \in N$, $\alpha \in T(E \cup N)$, $q_0 \in Q$ is the initial state, and Q_f is a set of final states.

The language $L(G)$ generated by $\mathcal{LSAG}$ G, consists of all $t \in T(E)$ for which there exists a derivation tree such that its root is labelled with the pair (q_0, S) and each leaf is labelled with a pair (p, t), where $p \in Q_f$. The membership value of t in $L(G)$ is calculated by a method similar to that of a language generated by an $\mathcal{LAG}$.

A derivation tree $\mathcal{T} \in T(Q, T(E \cup N))$ is defined as a partial function $\mathcal{T} : \mathbb{N}^* \to (Q, T(E \cup N))$, with domain $pos(\mathcal{T})$ defined similar to Definition 3.3.

Remark 4.4. If $(p, w) \to^{c_j} (q_j, \alpha_j)$, where $w \in N$, $p, q_j \in Q$ and $j = 1, \cdots, k$, are all the productions in R that have (p, w) as a left-hand side such that $(p, f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n))$ is a tree in $\mathcal{T}(p)$, then:

- if p is a universal state, then by applying all of the productions $(p, w) \to^{c_j} (q_j, \alpha_j)$, $\mathcal{T}(p)$ will have k sons $((q_1, f(u_1, \cdots, u_{i-1}, \alpha_1, u_{i+1}, \cdots, u_n)), \cdots, (q_k, f(u_1, \cdots, u_{i-1}, \alpha_k, u_{i+1}, \cdots, u_n)))$, with membership value $(c_1, \cdots, c_k)$,
- if p is an existential state, then by applying one of the productions $(p, w) \to^{c_j} (q_j, \alpha_j)$, $j = 1, \cdots, m$, $\mathcal{T}(p)$ will have a single son.

An $\mathscr{L}$-valued state alternating normalized regular tree grammar ($\mathcal{LSANG}$) $G = (Q, U, S, N, E, R, q_0, Q_f)$ is an $\mathcal{LSAG}$ in which the productions are in the form $(q, A) \to^c (q', a)$, $(q, A) \to^c (q', f(A_1, \cdots, A_n))$ or $(q, A) \to^c (q', A')$ where f and a are symbols of E and $A_1, \cdots, A_n, A, A'$ are non-terminals. We can easily show that $\mathcal{LSANG}$ and $\mathcal{LSAG}$ are equivalent.

The following two lemmas show that $\mathcal{LEAG}$s and $\mathcal{LSAG}$s are equivalent.

Lemma 4.5. *For each $\mathcal{LEAG}$ G, there exists an $\mathcal{LSAG}$ G' such that $L(G) = L(G')$.*

Proof. Let $G = (Q, S, N, U, E, R, q_0, Q_f)$ be an $\mathcal{LEAG}$. We define an $\mathcal{LSAG}$ $G' = (Q' \cup Q'', Q'', S, N, E, P, q_0', Q_f')$ as follows:

- $Q' = \{q' | q \in Q\}$,
- $Q_f' = \{q' | q \in Q_f\}$,
- $Q'' = \{q'' | q \in Q\}$, and
- P is as follows:
  - (i) for $w \in N \backslash U$, if $((p, w) \to^c (q, \alpha)) \in R$, where $p, q \in Q$ and $\alpha \in T(E \cup N)$, then $((p', w) \to^c (q', \alpha)) \in P$,
  - (ii) for $w \in U$, if $((p, w) \to^c (q, \alpha)) \in R$, where $p, q \in Q$ and $\alpha \in T(E \cup N)$, then $(p', w) \to^{c_1} (p'', w)$ and $(p'', w) \to^{c_2} (q', \alpha)$ are included in P, where $c = c_1 \wedge c_2$.

The root of any derivation tree $\mathcal{T}$ of G is labelled with the pair (q_0, S), while the root of any derivation tree $\mathcal{T}'$ of G' is labelled with the pair (q_0', S). Let z be a node of $\mathcal{T}$ with label $(p, f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n))$, where $p \in Q$, $u_i \in T(E \cup N)$ and $w \in N$, and assume that $\mathcal{T}'$ contains a corresponding node z' with label $(p', f(u_1, \cdots, u_{i-1}, w, u_{i+1} \cdots, u_n))$.
If $w \in N \backslash U$, then

$$(p, f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n)) \Rightarrow^c (q, f(u_1, \cdots, u_{i-1}, \alpha, u_{i+1}, \cdots, u_n)),$$

where $((p, w) \rightarrow^c (q, \alpha)) \in R$. Consequently, the node z' of $\mathcal{T}'$ will get a son z'' with label

$$(q', f(u_1, \cdots, u_{i-1}, \alpha, u_{i+1}, \cdots, u_n)),$$

that is,

$$(p', f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n)) \Rightarrow^c$$
$$(q', f(u_1, \cdots, u_{i-1}, \alpha, u_{i+1}, \cdots, u_n)),$$

where $((p', w) \rightarrow^c (q', \alpha)) \in P$.
Now, if $w \in U$, then

$$(p, f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n)) \Rightarrow^{(c_1, \cdots, c_k)}$$
$$((q_1, f(u_1, \cdots, u_{i-1}, \alpha_1, u_{i+1}, \cdots, u_n), \cdots,$$
$$(q_k, f(u_1, \cdots, u_{i-1}, \alpha_k, u_{i+1}, \cdots, u_n))),$$

where $((p, w) \rightarrow^{c_j} (q_j, \alpha_j)) \in R, 1 \leq j \leq k$.
Accordingly,

$$(p', f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n)) \Rightarrow^c$$
$$(p'', f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n)) \Rightarrow^{(c_1', \cdots, c_k')}$$
$$((q_1', f(u_1, \cdots, u_{i-1}, \alpha_1, u_{i+1}, \cdots, u_n)), \cdots,$$
$$(q_k', f(u_1, \cdots, u_{i-1}, \alpha_k, u_{i+1}, \cdots, u_n))),$$

where $(p', w) \rightarrow^c (p'', w), (p'', w) \rightarrow^{(c_1', \cdots, c_k')} (q_j', \alpha_j)$ and $c_j = c \wedge c_j', j = 1, \cdots, k$ are included in P. Hence, $L(G) \subseteq L(G')$.

Conversely, let $(t, c) \in L(G')$. Then, there exists a derivation tree $\mathcal{T}'$ of G' such that its root is labelled with (q_0', S) and each leaf is labelled with (q', t) for some $q \in Q_f$. By (i) and (ii), for each production in P, there exists a corresponding production in R. By applying these productions in derivation, it is easy to see that from $\mathcal{T}'$ we obtain a derivation tree $\mathcal{T}$ of G such that its root is labelled with (q_0, S) and each leaf is labelled with (q, t) for some $q \in Q_f$, that is, $L(G') \subseteq L(G)$. Hence $L(G) = L(G')$. □

Lemma 4.6. *For each $\mathcal{LSAG}$ G, there exists an $\mathcal{LEAG}$ G' such that $L(G) = L(G')$.*

Proof. Let $G = (Q, U, S, N, E, R, q_0, Q_f)$ be an $\mathcal{LSAG}$. Then, we define an $\mathcal{LEAG}$ G' as follows:

$$G' = (Q \cup Q', S'', N' \cup N'', N', E, P, q_0, Q_f),$$

where

$$Q' = \{q' \mid q \in U\},\ N' = \{w' \mid w \in N\},\ \text{and}\ N'' = \{w'' \mid w \in N\}.$$

Also, let $\varphi'' : T(E \cup N) \to T(E \cup N'')$ be a morphism which replaces each variable w with the new variable w''. The set of production rules, P, is defined as follows:

(1) for $q \in Q \backslash U$, if $((q, w) \to^c (p, \alpha)) \in R$, where $w \in N, p \in Q$ and $\alpha \in T(E \cup N)$, then $((q, w'') \to^c (p, \varphi''(\alpha))) \in P$,
(2) for $q \in U$, if $((q, w) \to^c (p, \alpha)) \in R$, where $w \in N, p \in Q$ and $\alpha \in T(E \cup N)$, then $((q, w'') \to^{c_1} (q', w')) \in P$ and $((q', w') \to^{c_2} (p, \varphi''(\alpha))) \in P$, where $c = c_1 \wedge c_c$,

By a method similar to that of Lemma 4.5, we can conclude that $L(G) = L(G')$. □

Now, from Lemmas 4.5 and 4.6, we obtain the following theorem.

Theorem 4.7. *The classes $\mathcal{LEAG}s$ and $\mathcal{LSAG}s$ are equivalent.*

Thus, from now on we will not discuss $\mathcal{LEAG}s$ anymore, but consider $\mathcal{LSAG}s$ instead. In Figure 5 we show the relation between various generalizations of $\mathscr{L}$-valued alternating regular tree grammars.

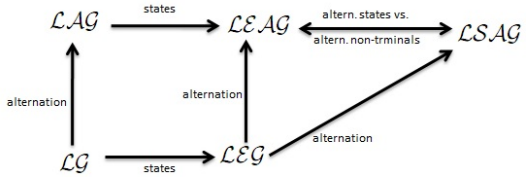

FIGURE 5. Taxonomy of $\mathscr{L}$-valued Alternating Regular Tree Grammars

Example 4.8. Let $\mathscr{L} = [0, 1]$ and $G = (Q, S, N, U, E, R, q_0, Q_f)$ be an $\mathcal{LEAG}$, where

$$Q = \{q_0, q_1, q_2, q_3, p_1, p_2, p_3\},\ N = \{S, A, B, A', B'\},$$
$$U = \{S\},\ E = \{a, b, g(), f(,)\},\ Q_f = \{q_3, p_3\},$$

and R consists of the following production rules:

$$(q_0, S) \to^{0.2} (q_1, f(A, B)),\ (q_0, S) \to^{0.7} (p_1, f(A', B')),$$
$$(q_1, A) \to^{0.5} (q_1, a),\ (q_1, A) \to^{0.5} (q_2, b),$$
$$(q_1, B) \to^{0.3} (q_1, g(B)),\ (q_1, B) \to^{0.4} (q_3, b)$$
$$(q_2, B) \to^{0.8} (q_3, g(B))),\ (q_3, B) \to^{0.3} (q_3, g(B)),$$
$$(q_3, B) \to^{0.3} (q_3, a),\ (p_1, A') \to^{0.6} (p_1, a),$$
$$(p_1, A') \to^{0.5} (p_2, b),\ (p_1, B') \to^{0.2} (p_1, g(B')),$$
$$(p_1, B') \to^{0.5} (p_3, b)\ (p_2, B') \to^{0.3} (p_3, g(B')),$$
$$(p_3, B') \to^{0.4} (p_3, g(B')),\ (p_3, B') \to^{0.1} (p_3, a).$$

Then, the derivation trees in G must be of the forms given in Figures 6, 7, 8, and 9, where $n, k \geq 0$ and $m, h \geq 1$.

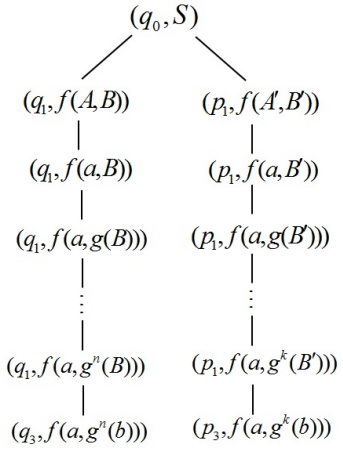

FIGURE 6. A Derivation Tree of Example 4.8

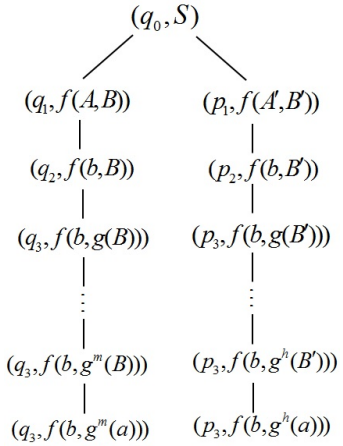

FIGURE 7. A Derivation Tree of Example 4.8

It is obvious that, $L(G) = \{(f(a, g^n(b)), 0.2), (f(b, g^m(a)), 0.2) |\ n \geq 0,\ m \geq 1\}$ is the language generated by G.
Now, we define the $\mathcal{LSAG}$ $G' = (Q' \cup Q'', Q'', S, N, E, P, q'_0, Q'_f)$, where

$$Q' = \{q'_0, q'_1, q'_2, q'_3, p'_1, p'_2, p'_3\},\ Q'' = \{q''_0, q''_1, q''_2, q''_3, p''_1, p''_2, p''_3\},\ Q'_f = \{q'_3, p'_3\},$$

and P is as follows:

$$(q'_0, S) \to^1 (q''_0, S),\ (q''_0, S) \to^{0.2} (q'_1, f(A, B)),$$
$$(q'_1, A) \to^{0.5} (q'_1, a),\ (q'_1, A) \to^{0.5} (q'_2, b),$$
$$(q'_1, B) \to^{0.3} (q'_1, g(B)),\ (q'_1, B) \to^{0.4} (q'_3, b)$$

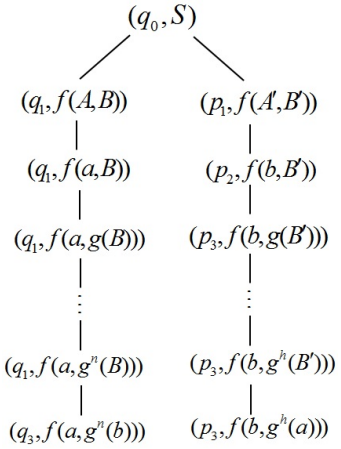

FIGURE 8. A Derivation Tree of Example 4.8

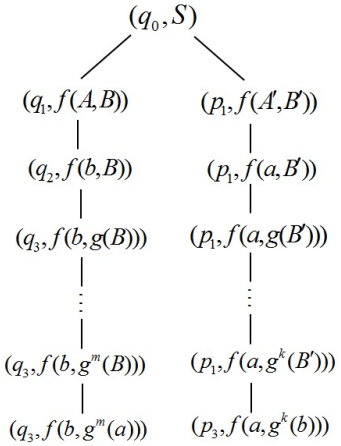

FIGURE 9. A Derivation Tree of Example 4.8

$$(q_2', B) \to^{0.8} (q_3', g(B))),\ (q_3', B) \to^{0.3} (q_3', g(B)),$$
$$(q_3', B) \to^{0.3} (q_3', a),\ (p_1', A') \to^{0.6} (p_1', a),$$
$$(p_1', A') \to^{0.5} (p_2', b),\ (p_1', B') \to^{0.2} (p_1', g(B')),$$
$$(p_1', B') \to^{0.5} (p_3', b)\ (p_2', B') \to^{0.3} (p_3', g(B')),$$
$$(p_3', B') \to^{0.4} (p_3', g(B')),\ (p_3', B') \to^{0.1} (p_3', a),$$
$$(q_0'', S) \to^{0.7} (p_1', f(A', B')).$$

Clearly, the language generated by G' is $L(G') = \{(f(a, g^n(b)), 0.2), (f(b, g^m(a)), 0.2) | n \geq 0,\ m \geq 1\}$. Thus $L(G') = L(G)$.

Conversely, let $G = (Q, U, S, N, E, R, q_0, Q_f)$ be an $\mathcal{LSAG}$, where

$$Q = \{q_0, q_1, q_2, p_1, p_2\},\ U = \{q_0\},\ N = \{S, A, B\},\ E = \{f(), a\},$$
$$Q_f = \{q_2, p_2\},$$

and R consists of the following productions:

$$(q_0, S) \to^{0.3} (q_1, f(A)),$$
$$(q_0, S) \to^{0.8} (p_1, f(B)),$$
$$(q_1, A) \to^{0.2} (q_1, f(A)),$$
$$(p_1, B) \to^{0.4} (p_1, f(B)),$$
$$(q_1, A) \to^{0.2} (q_2, a),$$
$$(p_1, B) \to^{0.8} (p_2, a).$$

It can be seen that $L(G) = \{f^n(a) |\ n \in N\}$. Now, we define the $\mathcal{LEAG}$ $G' = (Q \cup Q', S'', N' \cup N'', N', E, P, q_0, Q_f)$, where

$$Q' = \{q_0'\},\ N' = \{S', A', B'\},\ N'' = \{S'', A'', B''\},$$

and the production rules are:

$$(q_0, S'') \to^{1} (q_0', S'),$$
$$(q_0', S') \to^{0.3} (q_1, f(A'')),$$
$$(q_0', S') \to^{0.8} (p_1, f(B'')),$$
$$(q_1, A'') \to^{0.2} (q_1, f(A'')),$$
$$(p_1, B'') \to^{0.4} (p_1, f(B'')),$$
$$(q_1, A'') \to^{0.2} (q_2, a),$$
$$(p_1, B'') \to^{0.8} (p_2, a).$$

It is straightforward to see that $L(G') = \{f^n(a) |\ n \in N\}$. Thus $L(G) = L(G')$.

5. Basic Properties of $\mathcal{LAG}s$ and $\mathcal{LSAG}s$

The main goal of this section is to find a new tree automaton that accepts the language generated by $\mathcal{LSAG}$. Therefore, we will study some basic properties of the class of languages generated by $\mathcal{LAG}s$ and $\mathcal{LSAG}s$. In particular, we will see that the $\mathcal{LSAG}s$ are more powerful than the $\mathcal{LAG}s$ and we will establish a main result: there exists a variant of $\mathscr{L}$-valued alternating tree automaton that is equivalent to $\mathcal{LSAG}$. Since each $\mathcal{LAG}$ can be interpreted as an $\mathcal{LEAG}$ with only a single state, the following theorem can be obtained from Lemma 4.5 and show that any $\mathcal{LAG}$ can be simulated by an $\mathcal{LSAG}$.

Theorem 5.1. *For each $\mathcal{LAG}$ G, there exists an $\mathcal{LSAG}$ G' such that $L(G) = L(G')$.*

The above theorem shows that the class of $\mathcal{LSAG}s$ is at least as powerful as the class of $\mathcal{LAG}s$.

Example 5.2. Let $\mathscr{L} = [0,1]$ and $G = (S, N, U, E, R)$ be an $\mathcal{LAG}$, where

$$N = \{S, A, B, A', B'\},\ U = \{A\},\ E = \{f(,), g(), a, b\},$$

and R consists of the following productions:

$$S \to^{0.5} f(A, B),\ A' \to^{0.3} g(A'),\ B' \to^{0.7} g(B'),\ B \to^{0.5} b,$$
$$A' \to^{0.2} a,\ B' \to^{0.5} a,\ A \to^{0.8} A',\ A \to^{0.3} B'.$$

It is straightforward to verify that $L(G) = \{(f(g^n(a), b), 0.3) |\ n \geq 0\}$.
Now, we define the $\mathcal{LSAG}$ $G' = (\{q'_0, q''_0\}, \{q''_0\}, S, N, E, P, q'_0, \{q'_0\})$, where P contains the following productions:

$$(q'_0, S) \to^{0.5} (q'_0, f(A, B)),\ (q'_0, B') \to^{o.7} (q'_0, g(B')),\ (q'_0, A') \to^{0.3} (q'_0, g(A')),$$
$$(q'_0, A') \to^{0.2} (q'_0, a),\ (q'_0, B') \to^{0.5} (q'_0, a),\ (q'_0, B) \to^{0.5} (q'_0, b),$$
$$(q'_0, A) \to^{1} (q''_0, A),\ (q''_0, A) \to^{0.8} (q'_0, A'),\ (q''_0, A) \to^{0.3} (q'_0, B').$$

Clearly, $L(G') = \{(f(g^n(a), b), 0.3) |\ n \geq 0\}$. Consequently, $L(G) = L(G')$.

Problem. Does the converse of Theorem 5.1 hold? That is, can each $\mathcal{LSAG}$ be simulated by an $\mathcal{LAG}$?
As it has been shown, the language given in Example 3.17 can not be generated by any $\mathcal{LAG}$. In the following example, we show that it can be generated by an $\mathcal{LSAG}$. Therefore, the converse of Theorem 5.1 does not hold, in general. Therefore, the class of $\mathcal{LSAG}s$ is more powerful than the class of $\mathcal{LAG}s$.

Example 5.3. Consider the language given in Example 3.17. Now, we construct an $\mathcal{LSAG}$ $G = (Q, U, S, N, E, R, q_0, Q_f)$ as follows:
$Q = \{q_0, q_1, q_2, q_3, q_4\},\ U = \{q_1\},\ N = \{S, A, B, A', B'\},\ E = \{f(,), g(), a, b\}$,
$Q_f = \{q_4\}$, and the productions are as:

$$(q_0, S) \to^{0.5} (q_1, f(A, B)),\ (q_1, A) \to^{0.4} (q_2, A'),\ (q_1, A) \to^{0.5} (q_2, B'),$$
$$(q_2, A') \to^{0.3} (q_3, g(A')),\ (q_3, B) \to^{0.7} (q_2, g(B)),\ (q_2, B') \to^{0.2} (q_3, g(B')),$$
$$(q_2, A') \to^{0.5} (q_4, a),\ (q_2, B') \to^{0.6} (q_4, a),\ (q_4, B) \to^{0.3} (q_4, a).$$

It is obvious that $L(G) = \{(f(g^n(a), g^n(a)), 0.3) | n \geq 0\}$.

In the next theorem, it is shown that for linear grammars the converse of Theorem 5.1 holds.

Theorem 5.4. *For each linear $\mathscr{L}$-valued state alternating regular tree grammar ($lin - \mathcal{LSAG}$) G, there exists a linear $\mathscr{L}$-valued alternating regular tree grammar ($lin - \mathcal{LAG}$) G', such that G and G' generate the same language.*

Proof. Let $G = (Q, U, S, N, E, R, q_0, Q_f)$ be a $lin - \mathcal{LSAG}$. Then, we define a $lin - \mathcal{LAG}$ $G' = (S', N', U', E, R')$ as follows:

- $N' = \{\ [q, A] \mid q \in Q,\ A \in N\} \cup \{B\}$,
- $U' = \{\ [q, A] \mid q \in U,\ A \in N\}$,
- $S' = [q_0, S]$, and
- R' is obtained as follows:

(1) if $((p, A) \to^c (q, f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n))) \in R$, where $A, w \in N$, and $u_i \in T(E)$, then $([p, A] \to^c f(u_1, \cdots, u_{i-1}, [q, w], u_{i+1}, \cdots, u_n)) \in R'$,
(2) if $((p, A) \to^c (q, u)) \in R$, where $A \in N$, $u \in T(E)$, and $q \in Q_f$, then $([p, A] \to^c u) \in R'$,
(3) if $((p, A) \to^c (q, u)) \in R$, where $A \in N$, $u \in T(E)$, and $q \in Q \setminus Q_f$, then $([p, A] \to^c B) \in R'$.

Let $(t, c) \in L(G)$. Then, there exists a derivation tree $\mathcal{T}$ of G with the root labelled by (q_0, S) and each leaf labelled by (q, t) for some final state q. By replacing each node z with label $(q, f(u_1, \cdots, u_{i-1}, w, u_{i+1}, \cdots, u_n))$, where $q \in Q$, $u_i \in T(E)$ and $w \in N$, by a node z' with label $f(u_1, \cdots, u_{i-1}, [q, w], u_{i+1}, \cdots, u_n)$, and each node z with label (q, u), where $q \in Q_f$ and $u \in T(E)$, by a node z' with label u, we obtain a derivation tree $\mathcal{T}'$ from $\mathcal{T}$. Therefore $(t, c) \in L(G')$, that is, $L(G) \subseteq L(G')$. Also, it is easy to see that all trees generated by G' can also be generated by G, that is, $L(G') \subseteq L(G)$. Therefore, $L(G) = L(G')$. □

Theorems 5.1 and 5.4 yield the following consequence.

Corollary 5.5. (i) $L(\mathcal{LAG}) \subseteq L(\mathcal{LSAG})$,
(ii) $L(lin - \mathcal{LAG}) = L(lin - \mathcal{LSAG})$.

Example 5.6. Let $\mathscr{L} = [0, 1]$ and $G = (Q, U, S, N, E, R, q_0, Q_f)$ be a $lin - \mathcal{LSAG}$, where,

$$Q = \{q_0, q_1, q_2, p_1, p_2\},\ U = \{q_0\},\ N = \{S, A, B\},\ E = \{f(), a\},$$
$$Q_f = \{p_2, q_2\},$$

and R is as follows:

$$(q_0, S) \to^{0.6} (q_1, f(A)),$$
$$(q_0, S) \to^{0.3} (p_1, f(B)),$$
$$(q_1, A) \to^{0.4} (q_1, f(A)),$$
$$(q_1, A) \to^{0.1} (q_2, a),$$
$$(p_1, B) \to^{0.9} (p_2, a).$$

It is obvious that $L(G) = \{(f(a), 0.3)\}$.
Now, we construct the $lin - \mathcal{LAG}$ $G' = (S', N', U', E, R')$ with

$$S' = [q_0, S],\ N' = \{[q_0, S], \cdots, [p_2, B], B\},\ U' = \{[q_0, S], [q_0, A], [q_0, B]\},$$

and R' as follows:

$$[q_0, S] \to^{0.6} f([q_1, A]),$$
$$[q_0, S] \to^{0.3} f([p_1, B]),$$
$$[q_1, A] \to^{0.4} f([q_1, A]),$$
$$[q_1, A] \to^{0.1} a,$$
$$[p_1, B] \to^{0.9} a.$$

It can be seen that $L(G') = \{(f(a), 0.3)\}$. Thus $L(G) = L(G')$.

One of the problems stems from the relation $L(\mathcal{LAA}) \subseteq L(\mathcal{LSAG})$, is the following problem:
Problem. Is there a tree automaton $\mathcal{A}$, which accepts the languages generated by $\mathcal{LSAG}$?
The answer is positive. Therefore, next we introduce a new variant of $\mathscr{L}$-valued alternating tree automaton, the so-called $\mathscr{L}$-valued alternating stack tree automaton, or $\mathcal{LASA}$ for short.

Definition 5.7. An $\mathscr{L}$-valued alternating stack tree automaton $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, F, M, U, q_0,\ m_0, M_f, \lambda, \delta)$ where, where Q is a finite non-empty set of states, F is a ranked alphabet, M is a stack alphabet, $U \subseteq M$ is a set of universal stack symbols (symbols in $Q \setminus U$ are called existential stack symbols), $q_0 \in Q$ is the initial state, $m_0 \in M$ is the initial stack symbol, $m_f \subseteq Q$ is a final stack alphabet, λ is a set of $\mathscr{L}$-valued rules of the form $(m, q) \to^c (m', q')$ and δ is a set of $\mathscr{L}$-valued rules of the form:

$$(m, q(f(x_1, \cdots, x_n))) \to^c (m', f(q_1(x_1), \cdots, q_n(x_n))),$$

where $n \geq 0$, $m, m' \in M$, $f \in F_n$, $q, q', q_1, \cdots, q_n \in Q$.

Definition 5.8. Let $\mathcal{A} = (Q, F, M, U, q_0, m_0, M_f, \lambda, \delta)$ be an $\mathscr{L}$-valued alternating stack tree automaton . The relation $\to_{\mathcal{A}}$ is defined as:
assume $t, t' \in T(F \cup Q)$,

$$(m, t) \to_{\mathcal{A}} (m', t') \Leftrightarrow \begin{cases} \exists c \in C(F \cup Q), \exists u_1, \cdots, u_n \in T(F), \\ \exists (m, q(f(x_1, \cdots, x_n))) \to (m', f(q_1(x_1), \cdots, q_n(x_n))) \in \delta, \\ t = c[q(f(u_1, \cdots, u_n))], \\ t' = c[f(q_1(u_1), \cdots, q_n(u_n))]. \end{cases}$$

$\to_A^*$ is the reflexive and transitive closure of $\to_A$.

An instantaneous description (ID) of $\mathcal{LASA}$ on a tree $t \in T(F)$, computation tree and accepting computation tree of $\mathcal{LASA}$ can be defined as Definitions 3.2, 3.3 and 3.5 with a slight difference. The language $L(\mathcal{A})$ accepted by the $\mathcal{LASA}$ is defined as:

$$L(\mathcal{A}) = \{(t, \mu_A(t)) |\ (\exists m \in M_f)(m_0, q_0(t)) \to^* (m, t)\},$$

where, $\mu_A(t)$ (the membership value of t) is defined as Definition 3.5.

Lemma 5.9. *For each $\mathcal{LASA}$ $\mathcal{A}$, there exists an $\mathcal{LSAG}$ G such that $L(\mathcal{A}) = L(G)$.*

Proof. Let $\mathcal{A} = (Q, F, M, U, q_0, m_0, M_f, \lambda, \delta)$ be an $\mathcal{LASA}$. We define $G = (Q', U', S_{q_0}, N, E, R, q_{m_0}, Q'_f)$ as follows:

- $Q' = \{q_m |\ m \in M\}$,
- $U' = \{q_m |\ m \in U\}$,
- S_{q_0} is a new symbol,
- $N = \{A_q |\ q \in Q\}$,
- $E = F$,
- $Q'_f = \{q_m |\ m \in M_f\}$, and

- δ consists of the following rules:
  $(q_\alpha, A_q) \to^c (q_\beta, f(A_{q_1}, \cdots, A_{q_n})) \in R$ if and only if
  $$(\alpha, q(f(x_1, \cdots, x_n))) \to^c (\beta, f(q_1(x_1), \cdots, q_n(x_n))) \in \delta,$$
  $(q_\alpha, A_q) \to^c (q_\beta, A_{q'}) \in R$ if and only if $(\alpha, q) \to^c (\beta, q') \in \lambda$
  and
  $$(q_{m_0}, S_{q_0}) \to^1 (q_{m_0}, A_{q_0}) \in R.$$

The proof can be completed by induction on the length of derivation, similar to the proof of Lemma 3.12. □

Lemma 5.10. *For each $\mathcal{LSAG}$ G, there exists an $\mathcal{LASA}$ $\mathcal{A}$ such that $L(\mathcal{A}) = L(G)$.*

Proof. Let $G = (Q, U, S, N, E, R, q_0, Q_f)$ be an $\mathcal{LSANG}$. We construct an $\mathcal{LASA}$ $\mathcal{A} = (Q', F, M, U', q_S, \alpha_{q_0}, M_f, \lambda, \delta)$ as follows:

- $Q' = \{q_A | A \in N\}$,
- $F = E$,
- $M = \{\alpha_q |\ q \in Q\}$,
- $U' = \{\alpha_q | q \in U\}$,
- $M_f = \{\alpha_q |\ q \in Q_f\}$, and
- δ, λ consists of the following rules:
  $(q, A) \to^c (q', f(A_1, \cdots, A_n)) \in R$ if and only if
  $$(\alpha_q, q_A(f(x_1, \cdots, x_n))) \to^c (\alpha_{q'}, f(q_{A_1}(x_1), \cdots, q_{A_n}(x_n))) \in \delta,$$
  $(q, A) \to^c (q', a) \in R$ if and only if $(\alpha_q, q_A(a)) \to^c (\alpha_{q'}, a) \in \delta$,
  $(q, A) \to^c (q', A') \in R$ if and only if $(\alpha_q, q_A) \to^c (\alpha_{q'}, q_{A'}) \in \lambda$.

The proof can be completed by induction on the length of derivation. □

Combining Lemmas 5.9 and 5.10, we have the following theorem:

Theorem 5.11. *The class of $\mathcal{LSAG}$ languages is precisely the class of $\mathcal{LASA}$ languages.*

Thus, utilizing Theorem 5.11 we solve the problem of deciding for a given $\mathcal{LSAG}$ and an $\mathcal{LASA}$. We give an example for Lemma 5.10.

Example 5.12. Consider the $\mathcal{LSAG}$ G given in Example 5.3. Now we construct an $\mathcal{LASA}$ $\mathcal{A} = (Q', F, M, U', q_S, \alpha_{q_0}, M_f, \lambda, \delta)$ as follows:
$Q' = \{q_S, q_A, q_B, q_{A'}, q_{B'}\}$, $F = \{f(,), g(), a, b\}$, $M = \{\alpha_{q_0}, \alpha_{q_1}, \alpha_{q_2}, \alpha_{q_3}, \alpha_{q_4}\}$,
$U' = \{\alpha_{q_1}\}$, $M_f = \{\alpha_{q_4}\}$, and the rules are as:

$$(\alpha_{q_0}, q_s f(x_1, x_2)) \to^{0.5} (\alpha_{q_1}, f(q_A(x_1), q_B(x_2))),\ (\alpha_{q_1}, q_A) \to^{0.4} (\alpha_{q_2}, q_{A'}),$$

$$(\alpha_{q_2}, q'_A g(x)) \to^{0.3} (\alpha_{q_3}, g(q_{A'}(x))),\ (\alpha_{q_1}, q_A) \to^{0.5} (\alpha_{q_2}, q_{B'}),$$

$$(\alpha_{q_3}, q_B g(x)) \to^{0.7} (\alpha_{q_2}, g(q_B(x))),\ (\alpha_{q_2}, q_{A'} a) \to^{0.5} (\alpha_{q_4}, a),$$

$$(\alpha_{q_2}, q_{B'} g(x)) \to^{0.2} (\alpha_{q_3}, g(q_{B'}(x))),\ (\alpha_{q_4}, q_{B'} a) \to^{0.6} (\alpha_{q_4}, a),$$

$$(\alpha_{q_4}, q_B a) \rightarrow^{0.3} (\alpha_{q_4}, a).$$

It is obvious that $\mathcal{L}(\mathcal{A}) = \{(f(g^n(a), g^n(a)), 0.3) | n \geq 0\}$.

6. Conclusions

In this paper, we studied different types of $\mathcal{LAG}s$, and investigated the equivalence between them. We defined the concepts of $\mathcal{LAG}$ (Definition 3.6) and $\mathcal{LANG}$ (Definition 3.10) and showed the equivalence between them (Theorem 3.11). Furthermore, we proved that the languages accepted by $\mathcal{LAA}s$ are the same as the languages generated by $\mathcal{LAG}s$ (Theorem 3.14). Thereafter, we provided a tree language which was not acceptable by an $\mathcal{LAA}$ (Example 3.17). Therefore, we defined the concepts of $\mathcal{LEAG}$ (Definition 4.1) and $\mathcal{LSAG}$ (Definition 4.3) and proved the equivalence between them (Theorem 4.7). Furthermore, we proved that $L(\mathcal{LAG}) \subseteq L(\mathcal{LSAG})$ (Theorem 5.1), and by an example (Example 5.3), we showed that $L(\mathcal{LSAG}) \nsubseteq L(\mathcal{LAG})$, in general. However, we proved that for linear grammars, it follows that $L(lin - \mathcal{LAG}) = L(lin - \mathcal{LSAG})$ (Theorem 5.4). Also, we introduced an $\mathscr{L}$-valued alternating stack tree automaton (Definition 5.7) and showed that a language is acceptable by this automaton if and only if it can be generated by an $\mathcal{LSAG}$ (Theorem 5.11).
However, for future research, the notion of $\mathscr{L}$-valued alternating pushdown tree automata will be studied and some relationships with $\mathscr{L}$-valued alternating context free tree grammars will be tried.

Acknowledgements. The authors would like to express their sincere thanks to the editor and anonymous referees for their useful suggestions which improved the quality of the paper.

References

[1] A. Bouhoula, J. P. Jouannaud and J. Meseguer, *Specification and proof in membeship equational logic*, Theoretical Computer Science, **236** (2000), 35-132.
[2] G. Birkhoff, *Lattice theory*, American Mathematical Society Colloquium Publications, New York, 1984.
[3] S. Bozapalidis and O. L. Bozapalidoy, *Fuzzy tree language recognizability*, Fuzzy Sets and Systems, **161(5)** (2010), 716-734.
[4] J. R. Buchi, *Weak second-order arithmetic and finite automata*, Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik, **6** (1960), 66-92.
[5] Y. Cao, L. Xia and M. Ying, *Probabilistic automata for computing with words*, Journal of Computer and System Sciences, **79(1)** (2013), 152-172.
[6] A. K. Chandra, D. C. Kozen and L. J. Stockmeyer, *Alternation*, Journal of the ACM, **28(1)** (1981), 114-133.
[7] S. R. Chaudhari and M. N. Joshi, *A note on fuzzy tree automata*, International Journal of Computer Applications, **56(17)** (2012), 1-5.
[8] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Loding, S. Tison and M. Tommasi, *Tree automata: technigues and applications*, 2007. Available: http://tata.gforge.inria.fr.
[9] Z. Esik and G. Liu, *Fuzzy tree automata*, Fuzzy Sets and Systems, **158** (2007), 1450-1460.
[10] B. Finkbeiner and H. Sipma, *Checking finite traces using alternating automata*, Formal Methods in System Design, **24(2)** (2004), 101-127.
[11] F. Gecseg and M. Steinby, *Tree automata*, Akademiai Kiado, Budapest, 1984.

[12] M. Ghorani and M. M. Zahedi, *Characterization of complete residuated lattice-valued finite tree automata*, Fuzzy Sets and Systems, **199** (2012), 28-46.
[13] M. Ghorani, M. M. Zahedi and R. Ameri, *Algebraic properties of complete residuated lattice-valued tree automata*, Soft Computing, **16** (2012), 1723-1732.
[14] J. E. Hopcroft, R. Motwani and J. D. Ullman, *Introduction to automata theory, languages and computation*, 3rd edition, Addison-Wesley, 2006.
[15] H. Hosoya, J. Vouillon and B. C. Pierce, *Regular expression types for XML*, ACM Transactions on Programming Languages and Systems, **27(1)** (2005), 46-90.
[16] J. Ignjatovic, M. Ciric and S. Bogdanovic, *Determinization of fuzzy automata with membership values in complete residuated lattices*, Information Sciences, **178** (2008), 164-180.
[17] J. Jin, Q. Li and Y. Li, *Algebraic properties of L-fuzzy finite automata*, Information Science, **234** (2013), 182-202.
[18] D. Kirsten, *Alternating tree automata and parity games*, In: E. Gradel (Ed.), Automata, Logics, and Infinite Games, Springer-Verlag, Berlin, 2002.
[19] R. E. Ladner, R. J. Lipton and L. J. Stockmeyer, *Alternating pushdown automata*, Proceeding of 19th FOCS, IEEE Computer Society Press, Silver Spring, (1978), 92-106.
[20] R. E. Ladner, R. J. Lipton and L. J. Stockmeyer, *Alternating pushdown and stack automata*, SIAM Journal on Computing, **13** (1984), 135-155.
[21] E. T. Lee and L. A. Zadeh, *Note on fuzzy languages*, Information Sciences, **1** (1969), 421-434.
[22] H. X. Lei and Y. Li, *Minimization of states in automata theory based on finite lattice-ordered monoids*, Information Sciences, **177** (2007), 1413-1421.
[23] Y. M. Li and W. Pedrycz, *Minimization of lattice finite automata and its application to the decomposition of lattice languages*, Fuzzy Sets and Systems, **158(13)** (2007), 1423-1436.
[24] L. Li and D. Qiu, *On the state minimization of fuzzy automata*, IEEE Transaction on Fuzzy Systems, **23(3)** (2015), 434 - 443.
[25] Y. Li and Q. Wang, *The universal fuzzy automata*, Fuzzy Sets and Systems, **249** (2014), 27-48.
[26] F. Lin and H. Ying, *Modeling and control of fuzzy discrete event systems*, IEEE Trans. Syst., Man, Cybern. B, Cybern., **32** (2002), 408- 415.
[27] J. N. Mordeson and D. S. Malik, *Fuzzy automata and languages: theory and applications*, Chapman & Hall CRC, London, Boca Raton, 2002.
[28] E. Moriya, *A grammatical characterization of alternating pushdown automata*, Theoretical Computer Science, **67** (1989), 75-85.
[29] E. Moriya, D. Hofbauer, M. Huber and F. Otto, *On state-alternating context-free grammars*, Theoretical Computer Science, **337** (2005), 183-216.
[30] E. Moriya and F. Otto, *Two ways of introducing alternation into context-free grammars and pushdown automata*, IEICE Transactions on Information and Systems, **E90D(6)** (2007), 889-894.
[31] E. Moriya and F. Otto, *On alternating phrase-structure grammars*, In: C. Martin-Vide, F. Otto and H. Fernau (Eds.), Language and Automata Theory and Applications, Springer-Verlag Berlin, Heidelberg, 2008.
[32] C. W. Omlin, K. K. Thornber and C. L. Giles, *Fuzzy finite-state automata can be deterministically encoded in recurrent neural networks*, IEEE Trans. Fuzzy Syst., **5** (1998), 76-89.
[33] W. Pedrycz and A. Gacek, *Learning of fuzzy automata*, International Journal of Computational Intelligence and Applications, **1** (2001), 19-33.
[34] D. W. Qiu, *Automata theory based on completed residuated lattice-valued logic (I)*, Science in China (Series F), **44** (2001), 419–429.
[35] D. W. Qiu, *Automata theory based on completed residuated lattice-valued logic (II)*, Science in China (Series F), **45** (2002), 442–452.
[36] D. W. Qiu, *Characterizations of fuzzy finite automata*, Fuzzy Sets and Systems, **141** (2004), 391-414.
[37] D. W. Qiu, *Supervisory control of fuzzy discrete event systems: a formal approach*, IEEE Transactions on Systems, Man and Cybernetics-Part B, **35(1)** (2005), 72-88.

[38] D. W. Qiu, *Pumping lemma in automata theory based on complete residuated lattice-valued logic: a note*, Fuzzy Sets and Systems, **157** (2006), 2128-2138.
[39] E. S. Santos, *Maximin automata*, Inform. and Control, **12** (1968), 367-377.
[40] G. Slutzki, *Alternating tree automata*, In: G. Goos and J. Hartmanis (Eds.), 8th colloquium Laquila Proceeding on Trees in Algebra and Programming, Springer-Verlag, Berlin, 1983.
[41] G. Slutzki, *Alternating tree automata*, Theorical Computer Science, **41** (1985), 305-318.
[42] J. Tang, Y. Fang and J. G. Tang, *The lattice-valued Turing machines and the lattice-valued type 0 grammars*, Mathematical Problems in Engineering, **2014** (2014), 1-6.
[43] M. G. Thomason and P. N. Marinos, *Deterministic acceptors of regular fuzzy languages*, IEEE Trans. Syst., Man, Cybern., **4** (1974), 228-230.
[44] M. Y. Vardi, *Alternating automata and program verification*, In: J. Van Leeuwen (Ed.), Computer Science Today, Recent Trends and Developments, Springer-Verlag, Berlin, 1995.
[45] M. Y. Vardi, *An automata-theoretic approach to linear temporal logic*, In: F. Moller and G. Birtwistle (Eds.): Logics for Concurrency: Structure versus Automata, Springer-Verlag, Berlin, 1996.
[46] M. Y. Vardi, *Alternating automata: checking truth and validity for temporal logics*, Proceding of the 14th Int. Conference on Automated Deduction, Springer-Verlag, Berlin, 1997.
[47] K. N. Verma and J. Goubault-Larrecq, *Alternating two-way AC-tree automata*, Information and Computation, **205** (2007), 817-869.
[48] W. G. Wee and K. S. Fu, *A formulation of fuzzy automata and its application as a model of learning systems*, IEEE Trans. Systems Man Cybern., **5** (1969), 215-223.
[49] T. Wilke, *Alternating tree automata, parity games, and modal μ-calculus*, Bulletin of the Belgian Mathematical Society-Simon Stevin, **8(2)** (2001), 359-391.
[50] L. Wu and D. W. Qiu, *Automata theory based on completed residuated lattice-valued logic: reduction and minimization*, Fuzzy Sets and Systems, **161** (2010), 1635-1656.
[51] H. Y. Xing and D. W. Qiu, *Pumping lemma in context-free grammar theory based on complete residuated lattice-valued logic*, Fuzzy Sets and Systems, **160** (2009), 1141-1151.
[52] H. Y. Xing, D. W. Qiu and F. C. Liu, *Automata theory based on complete residuated lattice-valued logic: pushdown automata*, Fuzzy Sets and Systems, **160** (2009), 1125-1140.
[53] H. Y. Xing, D. W. Qiu, F. C. Liu and Z. J. Fan, *Equivalence in automata theory based on complete residuated lattice-valued logic*, Fuzzy Sets and Systems, **158** (2007), 1407-1422.

Maryam Ghorani*, Faculty of Mathematical Sciences, Shahrood University of Technology, Shahrood, Iran
E-mail address: ghorani@shahroodut.ac.ir

Mohammad Mehdi Zahedi, Department of Mathematics, Graduate University of Advanced Technology, Kerman, Iran
E-mail address: zahedi_mm@kgut.ac.ir

*Corresponding author