

Reasoning about co-Büchi Tree Automata*

Salvatore La Torre

Dipartimento di Informatica ed Applicazioni

Università degli Studi di Salerno

84081 Baronissi(SA), Italy

slatorre@unisa.it

Aniello Murano

Dipartimento di Scienze Fisiche

Università degli Studi di Napoli “Federico II”

I-80126 Napoli, Italy

murano@na.infn.it

Abstract. We consider co-Büchi tree automata along with both alternating and generalized paradigms, as a characterization of the class of languages whose complement is accepted by generalized Büchi tree automata. We first prove that for alternating generalized co-Büchi tree automata the simulation theorem does not hold and the generalized acceptance does not add to the expressive power of the model. Then, we show that the emptiness problem for this class is EXPTIME-complete. For the class of languages whose complement is accepted by deterministic generalized Büchi tree automata, we get better complexity bounds: we give a characterization of this class in terms of generalized co-Büchi tree automata that yields an algorithm for checking the emptiness that takes time linear in the product of the number of states and the number of sets in the acceptance condition. Finally, we compare the classes of languages whose complement is respectively accepted by deterministic and nondeterministic Büchi tree automata with the main classes studied in the literature.

Keywords: tree automata, emptiness problem, language complementation

Address for correspondence: Aniello Murano, Dipartimento di Scienze Fisiche, Università degli Studi di Napoli “Federico II”, Via Cinthia, Complesso Monte S. Angelo, I-80126 Napoli, Italy

*This research was supported by the MIUR grants 60% 2003-2004

1. Introduction

Since its early days the theory of automata had an astonishing impact in computer science. Several models of automata have been extensively studied and applied to many fields. In the sixties, with their pioneering work, Büchi [Büc62], McNaughton [McN66], and Rabin [Rab69, Rab70] enriched this theory by introducing finite automata on infinite objects. Such automata turn out to be very useful for those areas of computer science where nonterminating computations are studied. They give a unifying paradigm to specify, verify, and synthesize reactive systems [Kur94, VW86, VW94]. A system specification can be translated into an automaton, and thus, questions about systems and their specifications are reduced to decision problems in the automata theory. More precisely, given a system S and its specification φ , we can design an automaton A_S representing the system and an automaton $A_{\neg\varphi}$ accepting all computations that violate the specification. Thus, we can check the correctness of S with respect to φ by checking the emptiness of $A_S \times A_{\neg\varphi}$. The automata-theoretic approach separates the logical and the combinatorial aspects of reasoning about systems. The translation of specifications to automata handles the logic and shifts all the combinatorial difficulties to automata-theoretic problems. Recent industrial-strength property-specification languages such as Sugar [BBE⁺01], ForSpec [AFF⁺02], and the recent standard PSL 1.01 [www.accellera.org] include regular expressions and/or automata, making the automata-theoretic approach even more essential.

Automata on infinite objects come in two varieties: automata on infinite words and automata on infinite trees. In the modelling of nonterminating systems, computations can be seen as infinite sequences of system states and automata on infinite words are suitable to model such systems. To model nondeterminism, it is useful and natural to arrange computations in trees. It is worth noticing that some concurrent programs, such as operating systems, communication protocols, and air-traffic control systems, are intrinsically nondeterministic and nonterminating. Moreover, nondeterminism is successfully used to obtain models of concurrent programs in general (nondeterministic interleaving of atomic processes). Here we mainly consider automata on infinite trees. Automata on infinite words can be seen as automata on infinite trees with branching degree equal to one.

Automata on infinite trees recognize an input tree according to an acceptance criteria. In the literature, several acceptance criteria have been fruitfully investigated: recall *Büchi*, *co-Büchi*, *Muller*, *Rabin*, *Streett*, and *parity* acceptance conditions. This criteria have been first introduced on infinite words and then fruitfully extended to infinite trees (for more on these models see [GTW02]). Like automata on finite trees, an automaton on infinite trees either accept, or reject an input tree, in accordance with a run built as follows. Suppose that the automaton is in state q , is reading a node i of the input tree and suppose that i has branching degree n and is labelled with a . The pair (q, a) is mapped into a tuple of states (q_1, \dots, q_n) , in accordance with the transition rules of the automaton. The intuition is that the automaton splits into n copies, one at each child of i , and for each child j , it moves to state q_j , from which it should accept the subtree of the input tree rooted at j . Since a run on an infinite tree does not have final states, acceptance is determined with respect to the sets of states visited infinitely often during all paths of the run. For example, in the Büchi condition [Büc62, Rab70] a subset of the automaton states is defined as accepting and a tree is accepted if and only if there exists a run such that for each path “at least an accepting state repeats infinitely often”. The co-Büchi condition expresses the dual condition, that is, it requires that for each path “all states that are not accepting repeats finitely often” or equivalently “all the states that repeat infinitely often are accepting”. More general conditions are Muller, Rabin, Streett and parity automata, whose acceptance conditions involve several sets of states. For example, in Rabin

automata [Rab69] the acceptance condition is a set $\alpha = \{\langle G_1, B_1 \rangle, \dots, \langle G_k, B_k \rangle\}$ of pairs of sets of states, and a run is accepting if there is a pair $\langle G_i, B_i \rangle$ for which the set of states visited infinitely often intersects G_i and is disjoint from B_i . In parity automata [Mos84, EJ91], the acceptance condition is a set $\alpha = \{F_1, \dots, F_k\}$ of sets of states, and a run is accepting if the minimal index i for which the set F_i is visited infinitely often is even. Recall that Streett, Muller and parity automata are language equivalent to Rabin automata.

Büchi and co-Büchi conditions can be generalized in the sense that more than one subset of states can be defined as accepting. Thus, an infinite tree is accepted by a *generalized* Büchi automaton if and only if *for each* accepting set there is at least a state that repeats infinitely often. Consistently, an infinite tree is accepted by a *generalized* co-Büchi automaton if and only if *there is* an accepting set that contains all the states that repeat infinitely often. Generalized Büchi and co-Büchi acceptance conditions lead to automata with fewer states and simpler underlying structure than the corresponding standard conditions. For example, the traditional translation of an LTL formula φ to a Büchi word automaton results in an automaton with $2^{\mathcal{O}(|\varphi| \times |\varphi|)}$ states [VW94], while using generalized Büchi automata we only need $2^{\mathcal{O}(|\varphi|)}$ states [GPVW95]. Generalized conditions have become popular in system verification and now are fruitfully used in several applications [Kur94]. The generalized co-Büchi condition was first introduced and studied on infinite words in [Lan69]. Its extension to infinite trees has been investigated in [LMN02].

The kind of acceptance condition influences both the closure properties and the complexity of the decision algorithms. For generalized Büchi and generalized co-Büchi tree automata non-emptiness is decidable in polynomial time [VW86, LMN02], for Rabin tree automata it is known to be NP-complete [Rab69]. On the other hand, generalized Büchi and generalized co-Büchi tree automata are not closed under language complementation, while Rabin tree automata are [Tho90].

Another way to control the expressive power of automata is by restricting their transition relation to be *deterministic*. In a *nondeterministic* automaton, given a tree t and a node i on it labelled with a , the transition function maps the pair (q, a) into a (possibly empty) set of tuples of states (q_1, \dots, q_n) , where n is the branching degree of the node i . In a deterministic automaton, the transition function maps q and a into a single tuple of states. Thus, a deterministic automaton has a single run over an input tree, while a nondeterministic automaton may have many runs, and it accepts a tree if one of them is accepting. Reasoning about deterministic automata is usually simpler and is also motivated by the discovery that many natural specifications correspond to deterministic fragments: an LTL formula ψ has an equivalent alternation-free μ -calculus formula iff ψ can be recognized by a deterministic Büchi word automaton [KV98]. On the other side, nondeterministic automata are usually more expressive and succinct compared to the deterministic ones.

Nondeterminism can be viewed as if the automaton operates in an existential mode (a tree is accepted if there *exists* an accepting run). Dually, in a universal mode, a tree is accepted if *all* the runs of the automaton on it are accepting. In an *alternating automaton* [BL80, CKS81], both existential and universal modes are allowed and the transitions are given as Boolean formulas over sets of states. Thus, alternating automata generalize the standard notion of nondeterministic automata by allowing several successor states to go down along the same branch of the tree. The richer combinatorial structure of alternating automata have recently lead to their fruitful application in several industrial projects. For example, the Intel ForSpec compiler uses an intermediate language called SPIF, using alternating automata as temporal connectives. The ForSpec compiler translates ForSpec Temporal Logic formulas into SPIF, and from SPIF into nondeterministic Büchi automata [AFF⁺02].

Alternating tree automata along with the co-Büchi paradigm characterize the complement of lan-

languages nondeterministically accepted by Büchi tree automata [MS87]. Here, we consider these automata along with the generalized paradigm, namely, we consider alternating generalized co-Büchi tree automata (AGCTA), as a direct characterization of the class of languages whose complement is accepted by generalized Büchi tree automata (co-GBTA). In [MS87], it is shown that while alternation does not give more expressive power to Muller, parity, Rabin, Street and Büchi paradigms (*simulation theorem*), it allows us to get more succinct automata. For example, translating an alternating Büchi tree automaton to a Büchi tree automaton might involve an exponential blow-up [MS95]. Once the simulation theorem holds, emptiness for alternating automata can be checked in an easy, and often efficient, way via translation to the corresponding nondeterministic model. For example, for an alternating Büchi tree automaton, we can construct a language equivalent Büchi tree automaton (which involves an exponential blow-up) and thus we can check for emptiness the starting automaton in exponential time which matches the known lower bound for the computational complexity of this problem.

Here, we prove that unfortunately the simulation theorem does not hold for AGCTA. In fact, we observe that generalized co-Büchi tree automata are not sufficiently powerful to characterize co-GBTA. We also prove that AGCTA and alternating co-Büchi tree automata (ACTA) are polynomially time equivalent, that is, there exists a polynomial translation from an AGCTA to a language equivalent ACTA and viceversa. We observe that, when the generalized and the corresponding non-generalized paradigms are language equivalent, the generalized one is still of interest since it can lead to more succinct automata with evident benefits in designing efficient algorithms. As an example, we recall that nondeterministic generalized Büchi word automata and nondeterministic Büchi word automata are polynomially equivalent [Cho74]. However, computing the complement of a nondeterministic generalized Büchi automaton without constructing first the language equivalent nondeterministic Büchi automaton, may result in an automaton that is more succinct by an exponential factor [KV04].

Using the equivalence between AGCTA and ACTA, it follows that an AGCTA A can be translated to a parity tree automaton with two parity sets whose size is polynomial in the size of A . Thus, the emptiness problem for alternating generalized co-Büchi tree automata can be decided in exponential time. This result is also complete, since we can reduce the emptiness problem for weak alternating Büchi tree automata that is known to be EXPTIME-hard [KVW00].

To characterize the class of languages whose complement is accepted by generalized deterministic Büchi tree automata (co-DGBTA) we use the generalized co-Büchi paradigm along with the request that at least one path of an accepting run must be successful (\exists -*acceptance*). This kind of “existential” acceptance differs from the usual request for tree automata that all paths need to be successful in order to accept. With respect to the emptiness problem, this “existential” acceptance condition is equivalent to consider the tree automaton as a word automaton: each transition is split into several transitions (one for each state successor). Thus, given a DGBTA A with n states and k accepting sets, we can construct a Büchi word automaton B with $\mathcal{O}(nk)$ states such that the language accepted by B is empty if and only if the complement of the language accepted by A is empty. Using the fact that for Büchi word automata the emptiness problem is decidable in linear time [EL85], checking the emptiness for co-DGBTA can be decided in quadratic time. We recall that an elegant characterization of co-DGBTA can be obtained via weak alternating Büchi tree automata. Unfortunately, this characterization gives an exponential-time algorithm to solving the emptiness problem for co-DGBTA, since the emptiness problem for weak alternating Büchi tree automata is EXPTIME-complete [KVW00].

The rest of the paper is organized as follows. In Section 2, we give some basic definitions and recall some results of the theory of finite automata on infinite trees. In Section 3, we recall the concept of alter-

nation and the main properties of alternating tree automata. In Section 4, we consider alternation along with the generalized co-Büchi paradigm and compare the corresponding class of accepted languages with the main classes of languages considered in the literature. In Section 5, we deal with the class of languages whose complement is accepted by deterministic Büchi tree automata. Finally, we conclude with few remarks in Section 6.

2. Preliminaries

Let Σ be an alphabet, k be a positive integer, and $\text{DOM} = \{0, 1, \dots, k-1\}^*$. We define an infinite k -ary Σ -tree t as a map $t : \text{DOM} \rightarrow \Sigma$. The elements in DOM are the nodes of the tree, the empty word ε corresponds to the root and for each $w \in \text{DOM}$, wi is its i -child for $i \in \{0, 1, \dots, k-1\}$. In the following, unless differently stated, an infinite k -ary Σ -tree will be referred to simply as a tree. Let $u, v \in \text{DOM}$, we say that u precedes v , denoted as $u < v$, if there exists an x such that $v = ux$. Let $\pi \subseteq \text{DOM}$, π is a *path* of a tree t if it is a maximal subset of DOM linearly ordered by $<$. If π is a path of a tree t , then t/π denotes the restriction of the function t to the set π . We say that a symbol $a \in \Sigma$ occurs infinitely often in t/π if there exists an infinite number of nodes $u \in \pi$ such that $t(u) = a$. The set of symbols that occur infinitely often in t/π is denoted by $\text{Inf}(t/\pi)$.

Given a tree t and a node $u \in \text{DOM}$, we define the *subtree* of t rooted at u as the tree t_u such that $t_u(v) = t(uv)$, for $uv \in \text{DOM}$. Let Σ be a finite alphabet, we denote by T_Σ^ω the set of Σ -valued trees. A language is a subset of T_Σ^ω . Given a language $L \subseteq T_\Sigma^\omega$ we denote with \bar{L} the complement of L , that is, $\bar{L} = T_\Sigma^\omega \setminus L$. In the following, we deal exclusively with binary trees ($\text{DOM} = \{0, 1\}^*$). All the results we obtain also hold for k -ary trees, where $k \geq 2$. According to the introduced notation, we use t_0 and t_1 to denote, respectively, the subtrees of t rooted respectively at 0 and 1 (the two children of the root). Moreover, we notice that in a binary tree each path corresponds to a unique word in $\{0, 1\}^\omega$. In particular, with π_0 we denote the leftmost path 0^ω .

A *finite automaton on infinite trees* (TA) is a tuple $A = \langle \Sigma, Q, Q_0, \delta, F \rangle$ where Σ is the input alphabet, $Q \neq \emptyset$ is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $\delta \subseteq Q \times \Sigma \times Q \times Q$ is the transition relation, and F specifies the acceptance condition (a condition that defines a subset of Q^ω , we define several types of acceptance conditions below). Intuitively, each transition suggests a non-deterministic choice for the next configuration of the automaton. If $|Q_0| = 1$ and δ is a total function $\delta : Q \times \Sigma \rightarrow Q \times Q$, then A is a *deterministic automaton* (DTA). Given an input tree t , a *run* r of A on t is a Q -tree such that $r(\varepsilon) \in Q_0$, and $(r(u), t(u), r(u0), r(u1)) \in \delta$, for all $u \in \text{DOM}$. With $\text{Run}_A(t)$ we denote the set of runs of a TA A on a tree t . Clearly, if A is deterministic then $|\text{Run}_A(t)| = 1$. Different classes of languages are obtained defining different notions of *successful run*. A tree t is accepted by a TA A if there exists a successful run r of A on t , that is, r satisfies the acceptance condition on all the paths of t . We consider here the following conditions:

- a run r satisfies a *generalized Büchi* condition $F = \{F_1, \dots, F_k\} \subseteq 2^Q$ iff for each path π of r and for each set $F_i \in F$, $\text{Inf}(r/\pi) \cap F_i \neq \emptyset$;
- a run r satisfies a *generalized co-Büchi* condition $F = \{F_1, \dots, F_k\} \subseteq 2^Q$ iff for each path π of r there is a set $F_i \in F$ such that $\text{Inf}(r/\pi) \subseteq F_i$;
- a run r satisfies a *Muller* condition $F = \{F_1, \dots, F_k\} \subseteq 2^Q$ iff for each path π of r , $\text{Inf}(r/\pi) \in F$;

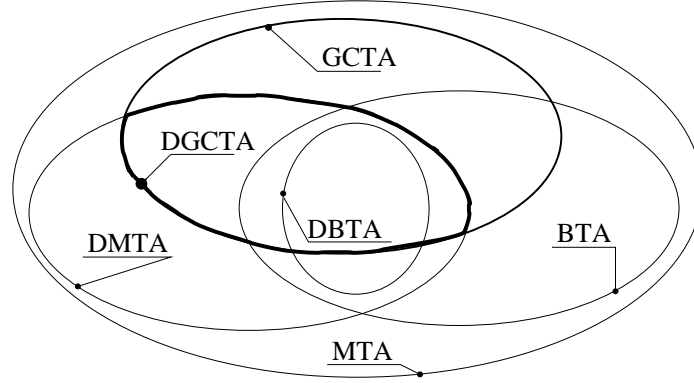


Figure 1. Relationships between (D)BTA, (D)MTA and (D)GCTA.

- a run r satisfies a *Rabin* condition $F = \{(B_1, G_1), \dots, (B_k, G_k)\} \subseteq 2^{Q \times Q}$ iff for each path π of r , there is a pair $(B_i, G_i) \in F$ such that $\text{Inf}(r/\pi) \cap B_i = \emptyset$ and $\text{Inf}(r/\pi) \cap G_i \neq \emptyset$;
- given a partition $F = \{F_1, F_2, \dots, F_{2k}\}$, $k \geq 1$, of the set of states, a run r satisfies the *parity* condition F iff for each path π of r the minimal index i for which $\text{Inf}(r/\pi) \cap F_i \neq \emptyset$ is even.

In the following, we refer to the number k appearing in the acceptance condition as the *index* of the corresponding automaton. Recall that Büchi and co-Büchi conditions are defined as the corresponding generalized conditions defined above with index 1. With $L(A)$ we denote the language accepted by a TA A , that is, the set of accepted trees.

To denote the different types of tree automata, we will use acronyms of the form DXTA and XTA, where X is one of B, GB, C, GC, M, R, P. The letter D stands for deterministic and the X is used to denote the kind of acceptance condition: Büchi (B), generalized Büchi (GB), co-Büchi (C), generalized Co-Büchi (GC), Muller (M), Rabin (R) and parity (P). For example, deterministic co-Büchi tree automata are denoted by DCTA. We also use the same acronyms to denote the corresponding class of accepted languages.

Figure 1 recalls the known relationships between all the considered classes of tree languages (automata) [Rab70, LMN02, Tho90, GTW02]. Since Rabin and parity conditions are equivalent to the Muller condition, the classes of languages accepted by the corresponding tree automata coincide in both the deterministic and nondeterministic paradigms. Thus, when we compare the class of languages we only refer to MTA and DMTA, and the obtained results clearly apply to the other classes. Analogously, since the class of languages (D)GBTA and (D)BTA coincide, in the language comparisons we only refer to (D)BTA.

The following theorem summarizes the closure properties of the above classes of automata and languages [Tho90, LMN02].

Theorem 2.1.

- DMTA, DGBTA, and DGCTA are closed under intersection, but they are not closed under union and complementation.

- GBTA and GCTA are closed under intersection and union, but they are not closed under complementation.
- MTA is closed under intersection, union, and complementation.

Given a tree automaton A , the *emptiness problem* for A is a well-known decision problem, used to establish if the language accepted by A is empty. Dually, the *universal problem* for A is used to decide if $L(A) = T_\Sigma^\omega$. In the following theorem, we recall some known results on the emptiness problems for Büchi, generalized co-Büchi, Rabin, and parity tree automata.

Theorem 2.2.

- The emptiness problem for BTA is decidable [Rab70], and is LOGSPACE-complete for PTIME[VW86].
- The emptiness problem for GCTA is decidable and is in PTIME [LMN02].
- The emptiness problem for PTA is $UP \cap \text{co-}UP$ [Jur98]¹.
- The non-emptiness problem for RTA is NP-complete [Rab69, EJ88].

3. Alternating tree automata

Alternating automata generalize the notion of nondeterminism by allowing several successor states along the same branch of the tree [MS87]. Muller and Schupp were the first to apply to tree automata the concept of alternation, introduced by Chandra, Kozen, and Stockmeyer [CKS81]. Here we briefly recall the basic definitions and refer to [MS95] for more details.

An *alternating tree automaton* is a TA with the transition relation defined as a function $\delta : Q \times \Sigma \rightarrow \mathcal{B}^+(K \times Q)$, where K is the set of *directions* in the tree ($K = \{0, 1\}$, for binary trees) and $\mathcal{B}^+(K \times Q)$ is the set of all positive boolean combinations of pairs (d, q) , where d is a direction and q is a state.

As an example, $\delta(q, a) = ((0, q_0) \vee (1, q_1)) \wedge (1, q_0)$ means that the automaton has two nondeterministic choices: one copy of the automaton proceeds to the 0-child of the current node entering state q_0 and another copy proceeds to the 1-child also entering state q_0 ; or two copies proceed to the 1-child entering respectively states q_1 and q_0 . Hence, \vee and \wedge in $\delta(q, a)$ represent, respectively, choice and concurrency.

A run of an alternating automaton on a binary tree t is a $(\{0, 1\}^* \times Q)$ -labeled (possibly non binary) tree such that the root is labelled (ε, q_0) and labels of each node and its successors must satisfy the transition relation δ . For example, if $t(\varepsilon) = a$ and $\delta(q_0, a) = ((0, q_1) \vee (0, q_2)) \wedge ((0, q_3) \vee (1, q_2))$, then, a run r on t at level 1 must include a node labeled $(0, q_1)$ or a node labeled $(0, q_2)$, and must include a node labeled $(0, q_3)$ or a node labeled $(1, q_2)$.

As for standard tree automata, we can couple different acceptance conditions to an alternating tree automaton, defining different classes of languages and automata. To denote alternating automata, we use a prefix “A” to the acronyms used so far. For example, we use ABTA to denote alternating Büchi tree automata, as well as the class of languages accepted by these automata.

In [MS87], it is shown that while alternation does not give more expressive power to Büchi, Muller, Rabin, Streett and parity paradigms, it gives more succinct automata as reported in the following theorem known in literature as (*simulation theorem*).

¹The class UP is a subset of NP, where each word accepted by the Turing machine has a unique accepting run

Theorem 3.1. (*Simulation theorem* [MS87]) There exists an effective procedure which, when given an alternating Büchi, Muller, Rabin, Streett or parity tree automata produces an equivalent nondeterministic Büchi, Muller, Rabin, Streett or parity tree automata, respectively. Moreover, the above procedure takes exponential time for Büchi automata and double exponential time in the other cases.

In [Cho74], it is shown that GBTA and BTA are polynomially equivalent. In the next lemma, we extend this result to the alternating paradigm. That is, given an AGBTA with m states and index k , we can build a language equivalent ABTA with $\mathcal{O}(m(k+1))$ states.

Lemma 3.1. Given an AGBTA A , there exists an ABTA A' accepting $L(A)$ and whose size is polynomial in the size of A .

Proof:

Let $A = \langle Q, \Sigma, \delta, Q_0, \{F_1, \dots, F_k\} \rangle$ be an AGBTA. Consider $A' = \langle Q \times \{0, \dots, k\}, \Sigma, \delta', Q_0 \times \{0\}, Q \times \{k\} \rangle$ as an ABTA, such that, for each formula $\delta(q, \sigma)$ in A , the automaton A' contains a formula $\delta'(q, i, \sigma)$ obtained from $\delta(q, \sigma)$ by coupling each pair (q', d) in $\delta(q, \sigma)$ with a value j as follows: (i) $j=0$ if $i = k$, (ii) $j = i + 1$ if $q \in F_j$, or (iii) $j = i$ otherwise. Thus, A' enters an accepting state if at least one state for all accepting sets from A has been visited infinitely often. Thus, $L(A') = L(A)$ and the size of A' is polynomial in the size of A . \square

Directly from the previous result we obtain the following.

Corollary 3.1. The simulation theorem holds for AGBTA.

In [MS87], Muller and Schupp introduced *weak alternating Büchi tree automata* (WABTA) as a special case for ABTA. In a WABTA, we have a Büchi acceptance condition $F \subseteq Q$ and there exists a partition of Q into disjoint sets, Q_1, \dots, Q_m , such that for each set Q_i , either $Q_i \subseteq F$, in which case Q_i is an *accepting* set, or $Q_i \cap F = \emptyset$, in which case Q_i is a *rejecting* set. In addition, there exists a partial order \leq on the collection of the Q_i 's such that for every $q \in Q_i$ and $q' \in Q_j$ for which $\delta(q, \sigma, q', q'')$ or $\delta(q, \sigma, q'', q')$ occurs, we have $Q_j \leq Q_i$. Thus, transitions from a state in Q_i lead to states in either the same set Q_i or in a lower one. It follows that every infinite path of a run of a WABTA ultimately gets “trapped” within some Q_i . The path then satisfies the acceptance condition if and only if Q_i is an accepting set.

The main properties about weak alternating Büchi, alternating Büchi, and alternating parity tree automata are summarized in the following theorem. We recall that an alternating automaton is deterministic if and only if the transition relation δ does not use \vee [MS95].

Theorem 3.2.

- Given an A(D)BTA (resp., an A(D)PTA) A , there exists a (D)BTA (resp., a (D)PTA) accepting $L(A)$, whose size is exponential in the size of A [MS87].
- The emptiness problem for (W)ABTA is EXPTIME-complete [KVV00, MS87].
- The emptiness problem for APTA is in EXPTIME [EJ91, Wil01].

Directly from Lemma 3.1 and Theorem 3.2 we also get the following result.

Corollary 3.2. The emptiness problem for AGBTA is decidable in exponential time.

As discussed in [MS87], an advantage of using alternation is that one can complement an alternating automaton by dualizing its transition function and acceptance condition. Formally, given a transition function δ , let $\tilde{\delta}$ denote the dual function of δ . That is, for every $\varphi \in \delta$, we have $\tilde{\varphi}$ in $\tilde{\delta}$, where $\tilde{\varphi}$ is obtained by φ switching \vee and \wedge and by switching *true* and *false*. The dual of an acceptance condition F , denoted as \tilde{F} , is a condition that holds exactly on all the runs on which F does not hold. In particular, by denoting with $\tilde{A} = \langle Q, \Sigma, \tilde{\delta}, Q_0, \tilde{F} \rangle$ the dual automaton of an automaton $A = \langle Q, \Sigma, \delta, Q_0, F \rangle$, the following holds.

Theorem 3.3. [MS87] For an ABTA A , the ACTA \tilde{A} accepts $\overline{L(A)}$, and viceversa.

4. Alternating generalized co-Büchi tree automata

In this section, we deal with alternating tree automata along with the generalized co-Büchi paradigm (AGCTA, for short). The definition of duality given in the previous section, along with the result shown in Theorem 3.3 makes this class a suitable choice for a direct characterization of the class of languages whose complement is in AGBTA, as pointed out in the following.

Corollary 4.1. Given an AGBTA A , its dual \tilde{A} is an AGCTA accepting $\overline{L(A)}$, and viceversa, given an AGCTA A , its dual \tilde{A} is an AGBTA accepting $\overline{L(A)}$.

Since A(G)CTA accepts the class of languages whose complement is accepted by (G)BTA, in the following, we also denote the class of languages accepted by A(G)CTA as co-(G)BTA. Lemma 4.1 shows that the class of languages accepted by AGCTA is polynomially equivalent to that accepted by ACTA.

Lemma 4.1. Given an AGCTA A , there exists an ACTA A' accepting $L(A)$ and whose size is polynomial in the size of A .

Proof:

Let A be an AGCTA, from Corollary 4.1, it follows that there exists an AGBTA B such that $\overline{L(B)} = L(A)$. From Lemma 3.1, it is possible to build an ABTA B' whose size is polynomial in the size of B such that $L(B) = L(B')$. From Theorem 3.3, there exists an ACTA A' dual to B' such that $\overline{L(B')} = L(A')$. Thus, $L(A') = \overline{\overline{L(B')}} = \overline{L(B)} = L(A)$ and the size of A' is polynomial in the size of A , from the definition of duality. \square

The above lemma is useful to show the following result.

Theorem 4.1. The emptiness problem for AGCTA is EXPTIME-complete.

Proof:

We first show that given an AGCTA A there exists an APTA B accepting $L(A)$ and whose size is polynomial in the size of A . From Lemma 4.1, we first translate A into an ACTA A' , whose size is polynomial in the size of A . Let $A' = \langle Q, \Sigma, \delta, Q_0, \{F\} \rangle$ be the obtained ACTA. An APTA accepting

$L(A')$ is the automaton $B = \langle Q, \Sigma, \delta, Q_0, \{Q \setminus F, F\} \rangle$. Thus, for the emptiness problem for AGCTA membership to EXPTIME follows from the fact that the size of B is linear in the size of A' , the size of A' is polynomial in the size of A , and the emptiness problem for APTA is in EXPTIME (see Theorem 3.2).

For the lower bound, we observe that each weak alternating Büchi tree automaton A can be translated into a language equivalent alternating co-Büchi tree automaton by simply interpreting its acceptance set as a co-Büchi condition. In fact, by the structure of the transition relation of a WABTA and the property that each set of the partition of its states is either contained into or disjoint from the acceptance set, we get that, along the paths of an accepting run of A , the states that repeat infinitely often are only states within the acceptance set. Since the emptiness problem for weak alternating Büchi tree automata is EXPTIME-hard [KVW00], we get that the emptiness problem for ACTA, and thus AGCTA, is EXPTIME-hard. \square

Directly from the above result, we also obtain the following.

Corollary 4.2. The universality problem for GBTA is EXPTIME-complete.

Proof:

The upper bound follows from Corollary 4.1 and from the fact that the emptiness problem for AGCTA is decidable in exponential time (Theorem 4.1). For the lower bound, we observe that the universality problem for automata on finite trees is EXPTIME-hard [Sei90]. \square

We now study the relationships between co-BTA and the classes of languages we have introduced in the previous sections. For this purpose, in Figure 2 we list some languages along with their ranking relatively to the classification illustrated in Figure 1. For more details see [LMN02]. For all these languages we assume that $\Sigma = \{a, b\}$. Since the classes of languages co-GBTA, co-BTA, AGCTA, and ACTA coincide, in the following we only use co-BTA to refer to this class.

Languages	Ranking
$L_1 = \{t \in T_\Sigma^\omega \mid \exists \pi, \text{ either } a \notin \text{Inf}(t/\pi) \text{ or } b \notin \text{Inf}(t/\pi)\}$	$(\text{GCTA} \cap \text{BTA}) \setminus \text{DMTA}$
$\overline{L}_1 = \{t \in T_\Sigma^\omega \mid \forall \pi, a \in \text{Inf}(t/\pi) \text{ and } b \in \text{Inf}(t/\pi)\}$	$\text{DBTA} \setminus \text{GCTA}$
$L_2 = \{t \in T_\Sigma^\omega \mid \forall \pi, \text{ either } a \notin \text{Inf}(t/\pi) \text{ or } b \notin \text{Inf}(t/\pi)\}$	$\text{DGCTA} \setminus \text{BTA}$
$\overline{L}_2 = \{t \in T_\Sigma^\omega \mid \exists \pi, a \in \text{Inf}(t/\pi) \text{ and } b \in \text{Inf}(t/\pi)\}$	$\text{BTA} \setminus \text{GCTA}$
$L_3 = \{t \in T_\Sigma^\omega \mid a \notin \text{Inf}(t/\pi_0)\}$	$(\text{BTA} \cap \text{GCTA} \cap \text{DMTA}) \setminus \text{DBTA}$
$\overline{L}_3 = \{t \in T_\Sigma^\omega \mid a \in \text{Inf}(t/\pi_0)\}$	DBTA

Figure 2. Some tree languages and their classification [LMN02].

Lemma 4.2. $\text{co-BTA} \not\subseteq \text{GCTA}$.

Proof:

This result can be shown using the languages L_1 and \overline{L}_1 . From the table in Figure 2, we know that $L_1 \in \text{BTA}$, thus $\overline{L}_1 \in \text{co-BTA}$, while $\overline{L}_1 \notin \text{GCTA}$. \square

Directly from the non-equivalence between generalized co-Büchi and alternating generalized co-Büchi paradigms shown in Lemma 4.2, we get the following important result for the latter.

Theorem 4.2. The simulation theorem does not hold for alternating generalized co-Büchi tree automata.

The following lemma states the results of all the remaining comparisons involving the class co-BTA. The complete picture of the relationships among all discussed classes is given in Figure 3.

Lemma 4.3.

1. $GCTA \subseteq \text{co-BTA}$.
2. $DMTA \subseteq \text{co-BTA}$.
3. BTA and co-BTA are not comparable.
4. $\text{co-BTA} \not\subseteq (GCTA \cup BTA \cup DMTA)$.
5. $(BTA \setminus (GCTA \cup DMTA)) \cap \text{co-BTA} \neq \emptyset$.
6. $BTA \cup \text{co-BTA} \subseteq MTA$.

Proof:

To prove that $GCTA \subseteq \text{co-BTA}$, we recall that any GCTA A is also an AGCTA, and thus, from Theorem 3.3, $L(A)$ is the complement of a language accepted by the AGBTA B dual of A . Thus, the result follows from the fact that the generalized Büchi paradigm is equivalent to Büchi. Strict containment is a consequence of Lemma 4.2. Thus part 1 holds.

To prove that $DMTA \subseteq \text{co-BTA}$, we first observe that on a tree t , a deterministic tree automaton can only check that the acceptance condition holds on a fixed path of t , or on all paths of t . Thus, given a DMTA M , $\overline{L(M)}$ consists of all trees such that the acceptance condition of M does not hold on a fixed path or on a nondeterministically selected path of t . Since a nondeterministic selection can be easily done in BTA, and since on a single path the deterministic Muller paradigm is equivalent to the nondeterministic Büchi one (see [Tho90]), we conclude that $\overline{L(M)}$ is in BTA, thus, $L(M)$ is in co-BTA. Moreover, since $GCTA \subseteq \text{co-BTA}$ and $DMTA \subseteq \text{co-BTA}$ but GCTA and MTA are not comparable (see Figure 1), we get that part 2 holds.

Part 3 follows directly from the non-closure under complementation of BTA (Theorem 2.1).

To prove part 4, we use the language $L = \{t \in T_\Sigma^\omega \mid t_0 \in L_1 \text{ and } t_{10} \in L_2 \text{ and } t_{11} \in \overline{L_1}\}$ (we recall that t_0, t_{10}, t_{11} are, respectively, the left subtree of t , the left subtree of t_1 and the right subtree of t_1). Since $L_1 \notin DMTA$, $L_2 \notin BTA$, and $\overline{L_1} \notin GCTA$ (see the table in Figure 2), we obtain that L is not in $(BTA \cup GCTA \cup DMTA)$. We conclude this part by noticing that \overline{L} is accepted by a BTA that nondeterministically checks that either t_0 is in $\overline{L_1}$, t_{10} is in $\overline{L_2}$, or t_{11} is in L_1 . Indeed, from table 2 we recall that $\overline{L_1}$, $\overline{L_2}$, and L_1 are in BTA.

To prove part 5, we use the language $L = \{t \in T_\Sigma^\omega \mid t_0 \in L_1 \text{ and } t_1 \in \overline{L_1}\}$. From the table in Figure 2 and the relationships among the introduced classes of languages shown in Figure 1, it follows that L is in $BTA \setminus (GCTA \cup DMTA)$. To prove that L is also in co-BTA we notice that \overline{L} is accepted by a BTA that nondeterministically checks that either t_0 is in $\overline{L_1}$, or t_1 is L_1 .

Finally, to prove part 6 we use the language $L = \{t \in T_\Sigma^\omega \mid t_0 \in \overline{L_2} \text{ and } t_1 \in L_2\}$. From the table in Figure 2 and from Figure 1, it follows that L is in MTA since L_2 is in DGCTA, $\overline{L_2}$ is in BTA and $DGCTA \cup BTA$ is in MTA. On the other hand, L is not in BTA since L_2 is not in BTA. Moreover, for L in co-BTA, \overline{L} must be in BTA, and thus, there must be a BTA that nondeterministically checks that

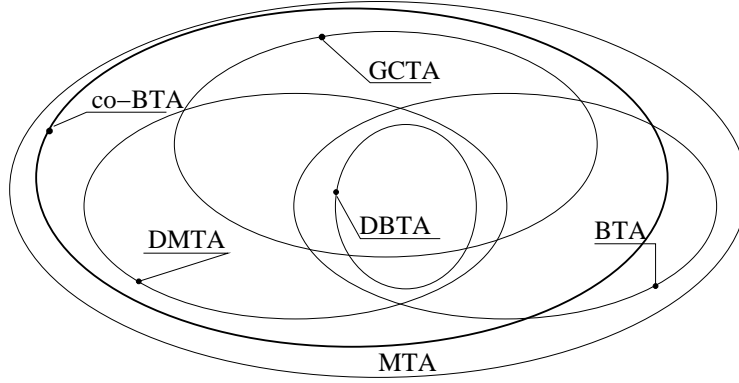


Figure 3. Summary of the comparisons involving co-BTA.

either t_0 is in L_2 or t_1 is in $\overline{L_2}$. From table in Figure 2, it is not possible, since L_2 is not in BTA. Hence, $L \in \text{MTA} \setminus (\text{BTA} \cup \text{co-BTA})$. \square

5. co-DGBTA

In this section, we deal with the class of languages whose complement is deterministically accepted by generalized Büchi tree automata (co-DGBTA, for short). We study the relationships of co-DGBTA with the other classes introduced so far and the complexity of the emptiness problem. Clearly, from the results obtained in the previous sections, the fact that DGBTA is polynomially equivalent to DBTA, and the fact that DBTA is a special case of BTA, it follows that the emptiness problem for co-DGBTA can be solved in exponential time. Here, we prove that this problem is indeed decidable in polynomial time.

The first example of class within BTA closed under complementation has been the remarkable characterization by Rabin [Rab72] of languages defined by a formula of weak monadic logic (where only quantifiers over finite sets are allowed): A language L is *weakly definable* if and only if both L and its complement \overline{L} are accepted by Büchi tree automata. In [MS87], it is shown that a language is weakly definable if and only if it is accepted by a weak alternating Büchi tree automaton. The class of languages accepted by these automata includes co-DBTA. In general, we have the following strict containment.

Lemma 5.1. $\text{DBTA} \cup \text{co-DBTA} \subset \text{WABTA}$.

Proof:

Let $L = \{t \in T_{\Sigma}^{\omega} \mid t_0 \in L_3 \text{ and } t_1 \in \overline{L_3}\}$, where L_3 and $\overline{L_3}$ are given in Figure 2. Since L_3 is in BTA and $\overline{L_3}$ is in DBTA, L is in BTA. On the other hand, L is not in DBTA, since L_3 is not in DBTA. Notice now that \overline{L} contains trees t such that either $t_0 \in \overline{L_3}$ or $t_1 \in L_3$. By the same reasoning as above, it follows that \overline{L} is in BTA and since L_3 is not in DBTA, also \overline{L} is not. Thus, L is also not in co-DBTA. Notice that since L and \overline{L} are in BTA, then L is accepted by a WABTA. Hence, $L \in \text{WABTA} \setminus (\text{DBTA} \cup \text{co-DBTA})$. \square

Recall that an alternating automaton is deterministic if and only if the transition relation δ does not use \vee [MS95]. Directly from this definition and from the fact that complementing a WABTA by

dualization gives an automaton with the same paradigm [MS87], we get the following characterization for co-DBTA.

Corollary 5.1. Each language in co-DBTA is accepted by a WABTA whose transition relation contains only disjunctions.

From Theorem 3.2, it follows that the emptiness problem for co-DBTA with the characterization given by Corollary 5.1 can be solved in exponential time. As we show in the following, this complexity can be reduced to a polynomial if we use a direct approach. First observe that a tree is not accepted by a DGBTA A if and only if the unique run r of A on t contains a path π that does not satisfy the acceptance condition. Thus, it is possible to characterize the complement of a language accepted by a DGBTA A with an automaton that, for each tree, nondeterministically selects a path and then deterministically checks that π does not satisfy the acceptance condition of A . This last corresponds to check that π satisfies the generalized co-Büchi condition obtained dualizing the acceptance condition of A . Thus, we modify the definition of accepting run. Given a tree t and a GCTA B with an accepting condition $F = \{F_1, \dots, F_k\}$, we say that a run $r \in \text{Run}_B(t)$ is \exists -successful if there exists a path π of r , such that $\text{Inf}(r/\pi) \subseteq F_i$ for some $F_i \in F$. A tree t is \exists -accepted by B if there exists an \exists -successful run of B on t . The language \exists -accepted by B is denoted by $L_{\exists}(B)$. In the next lemma, we show that the \exists -acceptance along with the generalized co-Büchi paradigm suffices to accept co-DGBTA.

Lemma 5.2. Given a DGBTA B , there exists a DGCTA A such that $L_{\exists}(A) = \overline{L(B)}$. Moreover, if B is DBTA then A is DCTA.

Proof:

Let L be a language whose complement is accepted by a DGBTA $B = \langle Q, \Sigma, \delta, Q_0, \{F_1, \dots, F_k\} \rangle$. Let $A = \langle Q, \Sigma, \delta, Q_0, \{Q \setminus F_i \mid i = 1, \dots, k\} \rangle$ be a DGCTA. A tree $t \notin L(B)$ if and only if the only run r in $\text{Run}_B(t)$ (B is deterministic) is not successful. That is, r contains at least a path π such that $\text{Inf}(r/\pi) \cap F_i = \emptyset$, for some i . Thus, by the definition of \exists -acceptance, $t \notin L(B)$ if and only if $t \in L_{\exists}(A)$. \square

With respect to the emptiness problem, notice that the characterization of co-DGBTA via “existential” tree automata is equivalent to consider tree automata as word automata². In more details, given a DGBTA B , consider a generalized co-Büchi word automaton C that is obtained from B by dualizing the acceptance condition and splitting each transition into several transitions, one for each state successor. That is, for each transition (s, σ, s', s'') of B , we get two transitions (s, σ, s') and (s, σ, s'') of C . It is easy to verify that there is a tree that is not accepted by B if and only if there is a word accepted by C . Using this observation, we get an efficient algorithm for solving the emptiness problem for co-DGBTA. First, we notice that a generalized co-Büchi word automaton can be easily translated into a language equivalent generalized Büchi word automaton whose size is linear in the size of the starting automaton. Thus, given a DGBTA B with n states and k accepting sets, we can construct a Büchi word automaton A with $\mathcal{O}(nk)$ states such that the language accepted by A is empty if and only if the complement of the language accepted by B is empty. Since the emptiness problem for Büchi word automata is decidable in

²Tree automata generalize word automata, in the sense that a word is a tree of arity 1. Thus, we omit a formal definition of word automata here.

linear time [EL85], we get that checking for the emptiness of $\overline{L(B)}$ can be done in $\mathcal{O}(nk)$ time. Thus, the following theorem holds.

Theorem 5.1. Given a DGBTA B with n states and index k , checking if $\overline{L(B)}$ is empty can be done in $\mathcal{O}(nk)$ time.

The rest of the section is devoted to compare co-DGBTA with the other classes considered in this paper.

Lemma 5.3. Given a DGBTA B , there exists a GCTA A such that $L(A) = \overline{L(B)}$. Moreover, if B is DBTA then A is CTA.

Proof:

Let L be a language whose complement is accepted by a DGBTA $B = \langle Q, \Sigma, \delta, Q_0, \{F_1, \dots, F_k\} \rangle$. We build a GCTA A that nondeterministically selects a path and on this path checks that the acceptance condition of B does not hold. Formally, $A = \langle \{q\} \cup Q, \Sigma, \delta', Q_0, \{\{q\} \cup Q \setminus F_i \mid i = 1, \dots, k\} \rangle$ be a GCTA, where $q \notin Q$ and δ' is defined as follows. For each $(s, \sigma, s', s'') \in \delta$, the transition relation δ' contains (s, σ, q, s'') and (s, σ, s', q) ; moreover, δ' contains (q, σ, q, q) . A tree $t \notin L(B)$ if and only if the only run r in $Run_B(t)$ (B is deterministic) is not successful. That is, r contains at least a path π such that $Inf(r/\pi) \cap F_i = \emptyset$, for some i . Thus, there exists a run r in $Run_A(t)$ such that for each path π , either $Inf(r/\pi) = \{q\}$ or there is an i such that $Inf(r/\pi) \subseteq Q \setminus F_i$. Hence, $t \notin L(B)$ if and only if $t \in L(A)$. \square

From the above construction, notice that co-DGBTA can be linearly characterized by GCTA. In the next lemma, we show that co-DGBTA can be polynomially characterized by BTA (notice that it is linear starting from co-DBTA).

Lemma 5.4. Given a DGBTA B , there exists a BTA A accepting $\overline{L(B)}$, whose size is polynomial in the size of B .

Proof:

By [Cho74], we can restrict to DBTA. Let L be a language whose complement is accepted by a DBTA $B = \langle Q, \Sigma, \delta, Q_0, F \rangle$. We build a BTA A that nondeterministically selects a path and on this path checks that the acceptance condition of B does not hold. Formally, $A = \langle Q', \Sigma, \delta', Q'_0, F' \rangle$ is such that (i) $Q' = Q \times \{0, 1, 2\}$; (ii) $Q'_0 = Q_0 \times \{0\}$; (iii) $F' = Q \times \{1, 2\}$; (iv) if $(s, \sigma, s', s'') \in \delta$, the transition relation δ' contains: $((s, 0), \sigma, (s', h), (s'', 1))$ and $((s, 0), \sigma, (s', 1), (s'', h))$, for $h \in \{0, 2\}$, $((s, 1), \sigma, (s', 1), (s'', 1))$, $((s, 2), \sigma, (s', 2), (s'', 1))$ for $s' \in Q \setminus F$, and $((s, 2), \sigma, (s', 1), (s'', 2))$ for $s'' \in Q \setminus F$. First, observe that size of A is linear in the size of B . Moreover, A accepts a tree t if and only if, for the only run r of B on t (B is deterministic), there exists a path π of r on which final states of B occur only finitely often. This is done by nondeterministically selecting a path π (unselected paths are marked with 1 in the second component of the states) and then checking that the property holds on π . For this purpose, on the selected path the second component of the states is nondeterministically set to 2. Once 2 is entered the run stops unless only states in $Q \setminus F$ are met on the selected path. Thus, $\overline{L(B)} = L(A)$, and the lemma is shown. \square

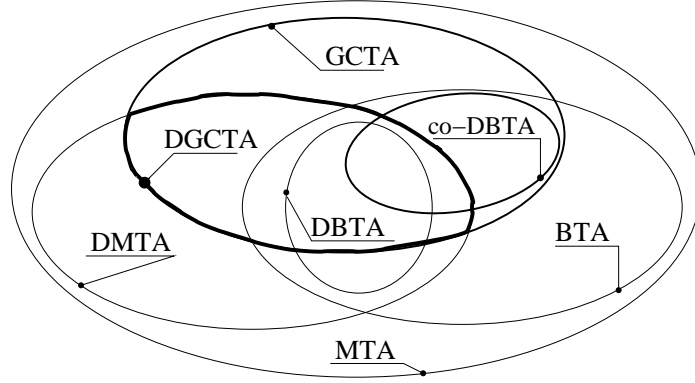


Figure 4. Summary of the comparisons involving co-DBTA.

Let us observe that the characterizations of co-DGBTA given in Lemmas 5.3 and 5.4 yield solutions to the emptiness problem for co-DGBTA via reductions to the same problem for GCTA and BTA, respectively. Since the best known upper bounds on the time complexity of the emptiness problem for GBTA and GCTA are both quadratic in the number of states and linear in the index of the automaton [Rab70, LMN02], the time complexity resulting from these approaches is asymptotically worse than the upper bound stated in Theorem 5.1.

In the following lemmas, we complete the comparisons involving co-DBTA and the classes GCTA, (D)BTA and DMTA. The complete picture of the comparisons is given in Figure 4.

Lemma 5.5. $\text{co-DBTA} \subseteq \text{GCTA} \cap \text{BTA}$.

Proof:

The inclusion $\text{co-DBTA} \subseteq \text{GCTA} \cap \text{BTA}$ is a direct consequence of Lemmas 5.3 and 5.4. To complete the proof, we only need to show that there exists a language in $(\text{BTA} \cap \text{GCTA}) \setminus \text{co-DBTA}$. Let L be the language defined as $\{t \in T_{\Sigma}^{\omega} \mid t_0 \in L_1 \text{ and } t_1 \in L_1\}$. Since $L_1 \in \text{BTA} \cap \text{GCTA}$, L is also in $\text{BTA} \cap \text{GCTA}$. To show that L is not in co-DBTA, we prove that the language $\bar{L} = \{t \in L \mid t_0 \in \bar{L}_1 \text{ or } t_1 \in \bar{L}_1\}$, corresponding to its complement, is not in DBTA. Assume that there exists a DBTA B accepting \bar{L} . Let $t \in \bar{L}$ such that $t_0 \in L_1$ and $t_1 \in \bar{L}_1$, and let r be a successful run from $\text{Run}_B(t)$. Consider now a tree t' obtained from t by exchanging the two subtrees t_0 and t_1 , that is $t'_0 = t_1$ and $t'_1 = t_0$. Obviously, $t' \in \bar{L}$. Denoting by r' a successful run of B on t' , replace t'_1 for t_1 in t , and call the obtained tree t'' . Clearly, $t'' \notin \bar{L}$. Since B is deterministic, and r and r' are both successful runs, we have that replacing r'_1 for r_1 in r , we obtain a successful run of B on t'' . Thus, $t'' \in L(B)$ and hence we get a contradiction. \square

Lemma 5.6.

1. (a) $\text{co-DBTA} \cap \text{DBTA} \neq \emptyset$;
 (b) $\text{co-DBTA} \cap (\text{DMTA} \setminus \text{DBTA}) \neq \emptyset$;
 (c) $\text{co-DBTA} \setminus \text{DMTA} \neq \emptyset$.
2. (a) $(\text{BTA} \cap \text{GCTA} \cap \text{DMTA}) \setminus \text{DBTA} \not\subseteq \text{co-DBTA}$;
 (b) $(\text{BTA} \cap \text{GCTA}) \setminus \text{DMTA} \not\subseteq \text{co-DBTA}$.

Proof:

Consider first part 1. To prove statement (a), we use the language $L = \{t \in T_\Sigma^\omega \mid \forall x \in \pi_0, t(x) = a\}$ and $\bar{L} = \{t \in T_\Sigma^\omega \mid \exists i \in \pi_0 \text{ such that } t(i) = b\}$. We first notice that both L and \bar{L} are in DBTA. Indeed the DBTA accepting L is $B = \langle \{q_a, q_b, q\}, \{a, b\}, \delta, \{q_a\}, \{q_a, q\} \rangle$ where $\delta = \{(q_a, a, q_a, q), (q_a, b, q_b, q), (q_b, x, q_b, q), (q, x, q, q) \mid x \in \Sigma\}$, while the DBTA accepting \bar{L} is the automaton B with acceptance condition $\{q_b, q\}$. Finally, from the fact that $L \in \text{DBTA}$ we also have that $\bar{L} \in \text{co-DBTA}$. To prove statement (b), we use L_3 which is in $\text{DMTA} \setminus \text{DBTA}$ and \bar{L}_3 which is in DBTA (see table in Figure 2). For statement (c) we use L_1 which is not in DMTA and \bar{L}_1 which is in DBTA (see table in Figure 2). Thus, L_1 is in co-DBTA.

Consider now part 2. To prove statement (a) we use $L = \{t \in T_\Sigma^\omega \mid t_0 \in L_3 \text{ and } t_1 \in L_3\}$. From the ranking of L_3 (see Figure 2), $L \in ((\text{BTA} \cap \text{GCTA} \cap \text{DMTA}) \setminus \text{DBTA})$. Notice that \bar{L} contains trees t such that either $t_0 \in L_3$ or $t_1 \in L_3$. From a reasoning analogously to that used in the proof of Lemma 5.5, it follows that \bar{L} cannot be accepted by a DBTA. Thus, $L \notin \text{co-DBTA}$. For statement (b) we use $L = \{t \in L \mid t_0 \in L_1 \text{ and } t_1 \in L_1\}$. Since $L_1 \in (\text{BTA} \cap \text{GCTA}) \setminus \text{DMTA}$, also L is in $(\text{BTA} \cap \text{GCTA}) \setminus \text{DMTA}$. \bar{L} contains trees t such that either $t_0 \in L_1$, or $t_1 \in L_1$. Thus, as before, \bar{L} is not in DBTA and, thus, $L \notin \text{co-DBTA}$.

6. Conclusion

Büchi and co-Büchi conditions are of interest for expressing requirements over nonterminating computations [GTW02]. For example, consider a drink-dispenser machine, we may want to express a requirement such as “users can always choose in the future coffee or tea” (typically a Büchi condition). In system verification, we may want to prove that the computations of a system do not violate a requirement. In particular, in the automata theoretic approach, given a system model S and its specification φ , we can construct an automaton A capturing the computations of S and an automaton B capturing the negation of φ . Thus, S is correct with respect to ψ if $L(A) \cap L(B)$ is empty [VW86, VW94]. In the above example, the negation of the assertion consists of requiring that “users can be prevented from choosing both coffee and tea from a given point on” (a co-Büchi condition). Thus, to prove a model A of the drink-dispenser correct with respect to the first requirement, we can model the second requirement as a tree automaton with co-Büchi acceptance and check if its intersection with A is empty.

In this paper, we have dealt with co-Büchi acceptance for branching time specifications. As a characterization of this class we have considered alternating generalized co-Büchi tree automata (AGCTA). We have compared the corresponding class of tree languages with the main classes of languages accepted by tree automata, showing interesting relationships. In particular, it is worth to remark that this class strictly contains the class accepted by co-Büchi tree automata and is not comparable with that characterized via Büchi tree automata. As a consequence of the first result we obtain that the simulation theorem does not hold for the co-Büchi acceptance condition on tree automata.

We have also investigated the emptiness problem for AGCTA and its sub-class of languages whose complement is accepted by deterministic generalized Büchi tree automata (co-DGBTA). For the general class, using a simple translation to parity automata, we have proved that the emptiness for AGCTA is in EXPTIME. This results is also complete since the emptiness problem for weak alternating Büchi tree automata is EXPTIME-hard. For the class co-DGBTA, we have shown a better bound, that is, the emptiness problem is decidable in quadratic time. For this purpose, we have used a linear-time

characterization of this class of languages via generalized co-Büchi tree automata. In particular, given a deterministic generalized Büchi tree automaton A with n states and index k , we can check the emptiness for the complement of $L(A)$ in time $\mathcal{O}(nk)$.

References

- [AFF⁺02] R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, E. Singerman, A. Tiemeyer, M.Y. Vardi, and Y. Zbar. The ForSpec temporal logic: A new temporal property-specification logic. In *Proc. 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of *Lecture Notes in Computer Science*, pages 296–211, Grenoble, France, April 2002. Springer-Verlag.
- [BBE⁺01] I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh. The temporal logic sugar. In *Proc. 13th International Conference on Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 363–367, Paris, France, July 2001. Springer-Verlag.
- [BL80] Janusz A. Brzozowski and Ernst L. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theor. Comput. Sci.*, 10:19–35, 1980.
- [Büc62] J.R. Büchi. On a decision method in restricted second-order arithmetic. In *Proceedings of the International Congress on Logic, Methodology, and Philosophy of Science 1960*, pages 1–12. Stanford University Press, 1962.
- [Cho74] Y. Choueka. Theories of automata on ω -tapes: a simplified approach. *Journal of Computer and System Sciences*, 8:117–141, 1974.
- [CKS81] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114 – 133, 1981.
- [EJ88] E.A. Emerson and C.S. Jutla. The complexity of tree automata and logics of programs. In *Proceedings 29th Annual IEEE Symp. on Foundations of Computer Science, FOCS'88*, pages 328 – 337, 1988.
- [EJ91] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings 32nd Annual IEEE Symp. on Foundations of Computer Science, FOCS'91*, pages 368–377, 1991.
- [EL85] E.A. Emerson and C.L. Lei. Modalities for model-checking: Branching time logic strikes back. In *Proceedings of the 12th ACM Symposium on Principles of Programming Languages*, pages 84–96, 1985.
- [GPVW95] R. Gerth, D. Peled, M.Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification Testing and Verification*, pages 3 – 18. Chapman & Hall, 1995.
- [GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*, 2002.
- [Jur98] Marcin Jurdziński. Deciding the winner in parity games is in $UP \cap co-UP$. *Information Processing Letters*, 68(3):119–124, 1998.
- [Kur94] R.P. Kurshan. *Computer-aided Verification of Coordinating Processes: the automata-theoretic approach*. Princeton University Press, 1994.
- [KV98] O. Kupferman and M.Y. Vardi. Freedom, weakness, and determinism: from linear-time to branching-time. In *Proc. 13th IEEE Symp. on Logic in Computer Science*, pages 81–92, June 1998.

- [KV04] O. Kupferman and M.Y. Vardi. From complementation to certification. In *10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, LNCS. Springer-Verlag, 2004.
- [KVV00] O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.
- [Lan69] L. H. Landweber. Decision problems for ω -automata. *Mathematical System Theory*, 3:376–384, 1969.
- [LMN02] S. La Torre, A. Murano, and M. Napoli. Weak muller acceptance condition for tree automata. In *3rd Workshop in Verification, Model Checking, and Abstract Interpretation, VMCAI 2002*. LNCS, volume 2294: 240-254, 2002.
- [McN66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.
- [Mos84] A. Mostowski. Regular expressions for infinite trees and a standard form of automata. 1984.
- [MS87] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.
- [MS95] D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata: new results and proofs of theorems of Rabin, McNaughton, and Safra. *Theoretical Computer Science*, 141:69–107, 1995.
- [Rab69] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1 – 35, 1969.
- [Rab70] M.O. Rabin. Weakly definable relations and special automata. *Mathematical Logic and Foundations of Set theory*, 1970.
- [Rab72] M.O. Rabin. Automata on infinite objects and church’s problem. *Trans. Amer. Math. Soc.*, 1972.
- [Sei90] H. Seidl. Deciding equivalence of finite tree automata. *SIAM Journal of Computing*, 19:424–437, 1990.
- [Tho90] W. Thomas. Automata on infinite objects. In J.van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol.B*, pages 133 – 191. 1990.
- [VW86] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:182 – 211, 1986.
- [VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 15:1 – 37, 1994.
- [Wil01] Thomas Wilke. Alternating tree automata, parity games, and modal μ -calculus. *Bull. Soc. Math. Belg.*, 8(2), May 2001.