Dresden University of Technology
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Complementation and Inclusion of Weighted Automata on Infinite Trees

Stefan Borgwardt Rafael Peñaloza

LTCS-Report 10-05

Complementation and Inclusion of Weighted Automata on Infinite Trees

Stefan Borgwardt Rafael Peñaloza

Abstract

Weighted automata can be seen as a natural generalization of finite state automata to more complex algebraic structures. The standard reasoning tasks for unweighted automata can also be generalized to the weighted setting. In this report we study the problems of intersection, complementation and inclusion for weighted automata on infinite trees and show that they are not harder than reasoning with unweighted automata. We also present explicit methods for solving these problems optimally.

1 Introduction

One of the current areas of interest in the field of automata theory is the study of weighted automata. These automata can be seen as a generalization of finite state automata in which the input structures are not accepted or rejected, but rather given a value called its *weight*. More formally, a weighted automaton defines a formal power series over a suitable algebraic structure [7].

The natural question to ask at the presence of such a generalization is whether the properties of the special case still hold. We can find several instances in the literature, where this question is answered affirmatively. For example, the relationship between automata and MSO logic, originally shown by Büchi [3], has been proven to hold also for weighted automata over finite and infinite words and trees [5, 8, 9, 16] and their corresponding weighted MSO logics. These results have motivated the development of automata-based procedures for multi-valued model checking [2]. For this purpose, standard reasoning tasks like deciding emptiness or complementing automata over finite or infinite words have been generalized to the weighted setting, and reasoning procedures solving these generalized problems have been developed. One interesting result obtained is that the complexity of these generalized reasoning tasks is not higher than in the unweighted case. For instance, the so-called *emptiness value problem* of these automata is NLogSpace-complete [11].

Despite reasoning with weighted automata on infinite *words* being well studied, there is a significant lack of results for weighted automata over infinite *trees*. In fact, to the best of our knowledge, the only explicit reasoning algorithm for these automata was given in [1], where a polynomial-time algorithm for computing the emptiness value of automata on infinite *unlabeled* trees, if the weights belong to a distributive lattice, is described. For labeled trees, a method that reduces the problem to several (unweighted) emptiness tests was described in [6]. The execution time of this approach, however, depends on the structure of the lattice.

In this paper we cover this gap by looking at reasoning problems for weighted automata on infinite trees that arise from generalizing standard problems for unweighted automata. In particular, we show that (weighted) union, intersection and emptiness of automata are computable in polynomial time, independently of the lattice used. We also look at the inclusion and complementation problems, and we show that their complexity remains equal to the unweighted case.

As for automata on infinite words, there are different kinds of automata on infinite trees (e.g., Büchi or co-Büchi automata; see Section 2.2). Since some of these classes are not closed under complementation, we analyze several different types of automata with their closure properties relative to each other. We also present explicit constructions for the complement of some classes of weighted and unweighted automata.

2 Automata on Infinite Trees

The main object of our study are weighted automata on infinite trees, whose weights belong to a distributive lattice [10]. We give a brief introduction to lattices before formally defining our automata models.

2.1 Lattices

A *lattice* is a partially ordered set $(S, \leq)$ where infima and suprema of arbitrary finite subsets of S always exist. The lattice $(S, \leq)$ is *finite* if its carrier set S is finite, it is *distributive* if the infimum and supremum operators distribute over each other, and it is *bounded* if it has a smallest element 0_S and a greatest element 1_S. In the following, we will often use the carrier set S to denote the lattice $(S, \leq)$. The infimum (supremum) of a finite subset $T \subseteq S$ will be denoted by $\bigotimes_{a \in T} a$ ($\bigoplus_{a \in T} a$). We will also use the infix notation if the set contains only two elements; i.e., $a \otimes b$ denotes the infimum of $\{a, b\}$.

A *Boolean lattice* is a bounded distributive lattice such that for every $a \in S$ there exists an element $\overline{a} \in S$ where $a \otimes \overline{a} = 0_S$ and $a \oplus \overline{a} = 1_S$. In this case, $\overline{a}$ is called the *complement* of a and is uniquely determined by these properties. Every

Boolean lattice is isomorphic to a powerset lattice $(\mathcal{P}(X), \subseteq)$, for some set X.

An element p of a lattice S is called *meet prime* if for every $a, b \in S$, $a \otimes b \leq p$ implies that either $a \leq p$ or $b \leq p$. The dual notion is that of a *join prime* element which is defined dually. Every element of a distributive lattice S can be computed as the infimum of all meet prime elements above it. In a Boolean lattice S, the complement $\overline{a}$ of a meet prime element $a \in S$ is join prime and vice versa. If the Boolean lattice S is isomorphic to the powerset lattice over some set X, then there are exactly $|X|$ meet prime and $|X|$ join prime elements in S.

In the following we will mainly deal with finite Boolean lattices. We now prove a few useful facts about these structures.

Lemma 1. *Let S be a finite Boolean lattice.*

a) For all $a, b \in S$, $\overline{a} \oplus b = 1_S$ iff $a \leq b$.

b) For every index set I and families $(f_i), (g_i) \in S^I$, the following holds:

$$\left(\bigoplus_{i\in I} f_i\right) \otimes \left(\bigotimes_{j\in I} g_j\right) \leq \bigoplus_{i\in I} (f_i \otimes g_i) \quad .$$

Proof. If $a \leq b$, then $\overline{a} \oplus b \geq \overline{a} \oplus a = 1_S$. Let now $\overline{a} \oplus b = 1_S$ and assume $a \not\leq b$. Then $\overline{b} \not\leq \overline{a}$ and thus $\overline{b} \neq \overline{b} \otimes \overline{a}$. But $(\overline{b} \otimes \overline{a}) \otimes b = 0_S$ and $(\overline{b} \otimes \overline{a}) \oplus b = \overline{a} \oplus b = 1_S$, which means that $\overline{b} \otimes \overline{a}$ is another complement of b. This contradicts the fact that the complement in S is unique.

For b), we have

$$\left(\bigoplus_{i\in I} f_i\right) \otimes \left(\bigotimes_{j\in I} g_j\right) = \bigoplus_{i\in I} \left(f_i \otimes \bigotimes_{j\in I} g_j\right) \leq \bigoplus_{i\in I} (f_i \otimes g_i)$$

by distributivity of S. □

2.2 Weighted Automata

We consider automata that receive as input infinite trees of a fixed arity k. For a positive integer k, we define $K := \{1, \ldots, k\}$. Our main object of study is the *full k-ary tree* K^*, where the root is denoted by the empty word ε, and the i-th successor of the node u is identified by ui. For a node u, we denote the full subtree of K^* with root u by $u[K^*]$.

Sometimes we will also speak about (finite) subtrees, i.e., finite prefix-closed subsets of $u[K^*]$.[1] A node in such a (finite) tree T is called *inner node* if all its

[1]Prefix-closed relative to u, i.e., up to and including u, but no prefix of u is considered.

successors are also elements of T. It is called *leaf* if it has no successors in T. The set of all inner nodes of T is called *interior* of T and is denoted by $\mathrm{int}(T)$. The set of all leaves of T is called *frontier* of T and is denoted by $\mathrm{fr}(T)$. T is called *closed* if $T = \mathrm{int}(T) \cup \mathrm{fr}(T)$.

A *path* p is a prefix-closed set of nodes such that if $u \in p$, then there is at most one $i \in K$ with $ui \in p$. $\mathcal{P}ath(u[K^*])$ denotes the set of all infinite paths in $u[K^*]$. A *labeled tree* is a mapping $t : u[K^*] \to \Sigma$ for some labeling alphabet Σ. As usual, the set of all such mappings is denoted by $\Sigma^{u[K^*]}$.

For an alphabet Σ and a lattice S, a *formal tree series over* Σ *and* S is a mapping $\Sigma^{K^*} \to S$; i.e., a function that maps each labeled tree to a value from S. For a formal tree series $f : \Sigma^{K^*} \to S$, the expression (f, t), called the *coefficient of* f *at* t, denotes the image of a tree t under f.

Definition 2. A *weighted generalized Büchi automaton (WGBA)* is a tuple $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt}, F_1, \ldots, F_n)$ where Q is a finite set of *states*, Σ is the *input alphabet*, S is a distributive lattice, $\mathrm{in} : Q \to S$ is the *initial distribution*, $\mathrm{wt} : Q \times \Sigma \times Q^k \to S$ is the *transition weight function* and $F_1, \ldots, F_n \subseteq Q$ are the sets of *final states*. A WGBA is called a *weighted Büchi automaton (WBA)* if $n = 1$ and *weighted looping automaton (WLA)* if $n = 0$.

A *run* of the WGBA $\mathcal{A}$ is a labeled tree $r \in Q^{K^*}$. This run is called *successful* if for every path $p \in \mathcal{P}ath(K^*)$ and every $i, 1 \leq i \leq n$ there are infinitely many nodes $u \in p$ such that $r(u) \in F_i$. The set of all successful runs of $\mathcal{A}$ is denoted by $\mathrm{succ}(\mathcal{A})$. We define the *transition* of r on $t \in \Sigma^{K^*}$ at a node $u \in K^*$ as $\overrightarrow{r(t,u)} := (r(u), t(u), r(ui), \ldots, r(uk))$. The *weight* of r on t is the value

$$\mathrm{wt}(t, r) := \mathrm{in}(r(\varepsilon)) \otimes \bigotimes_{u \in K^*} \mathrm{wt}(\overrightarrow{r(t,u)}) \ .$$

The *behavior* of $\mathcal{A}$ on an input tree $t \in \Sigma^{K^*}$ is

$$(\|\mathcal{A}\|, t) := \bigoplus_{r \in \mathrm{succ}(\mathcal{A})} \mathrm{wt}(t, r) \ .$$

Since the images of in and wt are finite, the infima and suprema in the above definitions are restricted to a finitely generated (and thus finite) distributive sublattice. Thus, even if S is an infinite lattice, the formal tree series $\|A\|$ has a finite image. In consequence, we can always assume w.l.o.g. that S is finite.

We will additionally consider *weighted co-Büchi automata (WCA)*. A WCA is like a WBA, except that a run is successful if every infinite path contains only finitely many states from $Q \setminus F_1$. Notice that WLA can be seen as special cases of both WBA and WCA in which $F_1 = Q$ and hence every run is successful.

In some of our proofs we will make use of known results for more expressive acceptance conditions that we now briefly describe. The *parity* condition is based on a priority function $Q \to \mathbb{N}$. A run is accepted if on every path the minimal priority occurring infinitely often is even. The *Streett* acceptance condition is based on pairs $(E_1, F_1), \ldots, (E_n, F_n)$ of state sets. A run is accepted if for every path $p \in \mathcal{P}\mathit{ath}(K^*)$ there is a pair (E_i, F_i) such that p contains infinitely many states from E_i or only finitely many states from F_i. The *Rabin chain* acceptance condition is also based on pairs $(E_1, F_1), \ldots, (E_n, F_n)$ where the strict inclusions $E_1 \subsetneq F_1 \subsetneq E_2 \subsetneq \ldots \subsetneq E_n \subsetneq F_n$ hold. A run is accepted if for every path $p \in \mathcal{P}\mathit{ath}(K^*)$ and every pair (E_i, F_i), p contains infinitely many states from F_i and only finitely many states from E_i.

The corresponding classes of automata are denoted by WPA, WSA and WRCA. For a given WBA or WCA it is easy to construct an equivalent WPA, WSA or WRCA. Also, parity, Streett and Rabin chain conditions can be reduced to each other by polynomial constructions. The class of tree series recognizable by WPA, WSA or WRCA strictly includes those recognizable by WBA or WCA. These in turn strictly include the tree series recognizable by WLA [18].

In the following we will often use expressions of the form WXA or XA as placeholders for the different acceptance conditions; i.e., WXA stands for a weighted automaton, and X is one of GB, B, C, L, P, S or RC. We will denote a generic weighted tree automaton as a tuple $(Q, \Sigma, S, \mathrm{in}, \mathrm{wt}, \mathfrak{F})$, where $\mathfrak{F}$ is the acceptance condition, i.e., it stands for several sets of states, pairs of sets of states or a priority function.

Standard (unweighted) automata can be seen as weighted automata over the lattice $\mathbb{B} := (\{0, 1\}, \leq)$, with $0 < 1$. The supremum and infimum in this lattice are denoted as $\vee$ and $\wedge$, respectively. The behaviour of such automata is the characteristic function of the set $\mathcal{L}(\mathcal{A}) := \{t \in \Sigma^{K^*} \mid (\|\mathcal{A}\|, t) = 1\}$ of all trees *accepted* by $\mathcal{A}$. Likewise, the functions in and wt can be seen as a set $I \subseteq Q$ and a relation $\Delta \subseteq Q \times \Sigma \times Q^k$, respectively. The abbreviations GBA, BA, CA, and LA will be used when the underlying lattice of the automaton is $\mathbb{B}$.

Definition 3. An *alternating tree automaton* is a tuple $\mathcal{A} = (Q, \Sigma, I, \delta, \mathfrak{F})$, where Q, Σ, I, and $\mathfrak{F}$ are defined as for (unweighted) tree automata. The *transition function* $\delta : Q \times \Sigma \to \mathcal{F}(Q \times K)$ maps each state and input symbol to a monotone Boolean formula over $Q \times K$.

Intuitively, an atomic formula (q, i) means that the automaton goes to state q at the i-th successor of the current node. Conjunction $\wedge$ means that the automaton splits up into several copies which each pursues the directions given by the conjuncts. Disjunction $\vee$ means that the automaton can make a non-deterministic choice as to which disjunct to follow.

Starting from the root and an initial state, from one starting automaton many copies can be generated, depending on the non-deterministic choices. Basically, each of these copies consists of a path taken through K^* and an associated sequence of states. An input tree is accepted if it is possible to make each of the non-deterministic choices in such a way that the state sequences generated by the resulting copies all satisfy the acceptance condition $\mathfrak{F}$.

Alternating tree automata are designated by the prefix A to the classification, e.g., ABA stands for the class of all alternating automata with a Büchi acceptance condition.

Example 4. A non-deterministic unweighted tree automaton $(Q, \Sigma, I, \Delta, \mathfrak{F})$ can easily be transformed into an alternating one by replacing Δ with the function

$$\delta(q, \alpha) := \bigvee_{(q,\alpha,q_1,\ldots,q_k)\in\Delta} \bigwedge_{i\in K} (q_i, i) \ ,$$

i.e., the automaton non-deterministically chooses a transition to take and then sends one copy in every direction.

2.3 Basic Results

A result that will be useful later is that to compute the behavior of a weighted automaton on a given input tree t it suffices to consider a finite subtree of K^*. We prove a more general result.

Lemma 5. *Let S be a finite lattice, Σ an input alphabet, $t \in \Sigma^{K^*}$ an input tree, Q a state set and $P : K^* \times (Q \times \Sigma \times Q^k) \to S$ a function that assigns a lattice element to each pair (u, y) consisting of a node and a transition. There is a closed, finite subtree $T \subseteq K^*$ such that for every run $r \in Q^{K^*}$ we have*

$$\bigotimes_{u\in K^*} P(u, \overrightarrow{r(t, u)}) = \bigotimes_{u\in \mathrm{int}(T)} P(u, \overrightarrow{r(t, u)}) \ .$$

Proof. We first construct the infinite tree R of all finite subruns. The root of R is labeled by the empty subrun $r : \emptyset \to Q$ and its direct successors are labeled with all subruns $r : \{\varepsilon\} \to Q$ of depth 0. For each node of R of depth n that is labeled with a subrun r of depth $n-1$, its successors are labeled with all extensions of r to subruns r' of depth n. Since r has k^{n-1} leaves, there are $k^{n-1}|Q|^k$ such extensions. Thus, R is finitely branching.

The tree R' is now constructed from R by pruning it as follows. We traverse R depth-first and check the label $r \in Q^T$ of each node. If there is an extension of r to a finite subrun $r' \in Q^{T'}$ with

$$\bigotimes_{u\in \mathrm{int}(T)} P(u, \overrightarrow{r(t, u)}) >_S \bigotimes_{u\in \mathrm{int}(T')} P(u, \overrightarrow{r'(t, u)}) \ ,$$

then we continue. Otherwise, we remove all nodes below the current node.

Since S is finite, for every run $r \in Q^{K^*}$ the expression $P(u, \overrightarrow{r(t,u)})$ can only yield finitely many different values. Thus, there must be a depth below which the value of the infimum of all $P(u, \overrightarrow{r(t,u)})$ is not changed anymore. Since every infinite path in R uniquely corresponds to a run $r \in Q^{K^*}$, this path must have been pruned in the construction of R', and thus R' can have no infinite paths.

Since R' is still finitely branching, by König's Lemma, R' must be finite and thus have a maximal depth m. Now it is easily seen that the tree $T := \bigcup_{n=0}^{m} K^n$ has the desired property. □

Note that this does not only hold for the infimum of the values $P(u, \overrightarrow{r(t,u)})$. Using the same arguments, an analogous result can be proven where $\bigotimes$ is substituted by $\bigoplus$.

Corollary 6. *For every weighted tree automaton $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt}, \mathfrak{F})$ with finite S and every input tree $t \in \Sigma^{K^*}$, there is a closed, finite subtree $T \subseteq K^*$ with the property that*

$$(\|\mathcal{A}\|, t) = (\|\mathcal{A}\|_T, t) := \bigotimes_{r \in \mathrm{succ}(\mathcal{A})} \mathrm{in}(r(\varepsilon)) \otimes \bigotimes_{u \in \mathrm{int}(T)} \mathrm{wt}(\overrightarrow{r(t,u)}) \ .$$

Proof. Apply Lemma 5 with $P(u, y) := \mathrm{wt}(y)$ for every $u \in K^*$ and $y \in Q \times \Sigma \times Q^k$. □

This means that the computation of $(\|\mathcal{A}\|, t)$ for a given t can be carried out in a finite amount of time, which is of course due to the finiteness of S. We now reformulate the above results for unweighted automata.

Corollary 7. *Let Σ be an input alphabet, $t \in \Sigma^{K^*}$ an input tree, Q a state set and $P \subseteq K^* \times (Q \times \Sigma \times Q^k)$ a predicate on pairs (u, y) of nodes and transitions. There is a closed, finite subtree $T \subseteq K^*$ such that for every run $r \in Q^{K^*}$ we have*

$$\forall_{u \in K^*} P(u, \overrightarrow{r(t,u)}) \iff \forall_{u \in \mathrm{int}(T)} P(u, \overrightarrow{r(t,u)}) \ .$$

□

Corollary 8. *For every unweighted tree automaton $\mathcal{A} = (Q, \Sigma, I, \Delta, \mathfrak{F})$ and every input tree $t \in \Sigma^{K^*}$, there is a closed, finite subtree $T \subseteq K^*$ with the property that*

$$t \in \mathcal{L}(\mathcal{A}) \iff \exists_{r \in \mathrm{succ}(\mathcal{A})} r(\varepsilon) \in I \wedge \forall_{u \in \mathrm{int}(T)} \overrightarrow{r(t,u)} \in \Delta \ .$$

□

Since weighted automata are a generalization of unweighted automata, a natural question is whether the standard results and constructions available for the latter can be adapted to the former. In unweighted automata, one is often interested in computing the union and intersection of the languages accepted by two automata. These operations correspond to a supremum and an infimum computation, respectively, from the lattice point of view. As the following lemma shows, these problems can be solved in polynomial time.

Lemma 9. *Let $\mathcal{A}$ and $\mathcal{B}$ be two WXA. Then one can construct automata $\mathcal{C}$ and $\mathcal{C}'$ of size polynomial on the sizes of $\mathcal{A}$ and $\mathcal{B}$ such that $(\|\mathcal{C}\|, t) = (\|\mathcal{A}\|, t) \otimes (\|\mathcal{B}\|, t)$ and $(\|\mathcal{C}'\|, t) = (\|\mathcal{A}\|, t) \oplus (\|\mathcal{B}\|, t)$ for all $t \in \Sigma^{K^*}$.*

Proof. These constructions closely follow the traditional constructions for intersection and union of finite automata, i.e., their state sets consist of the product and union of the original state sets, respectively. It is easy to combine the weight functions such that the desired behaviors are achieved. □

Another important problem for unweighted automata is deciding emptiness of the accepted language; i.e., whether there is at least one tree that is accepted. The natural generalization of this problem is to compute the supremum of the behaviour of $\mathcal{A}$ over all possible input trees. Using the ideas developed in [1], it is possible to show that this problem can be solved in polynomial time for WGBA (and hence also for WBA and WLA).

Lemma 10. *Given a WGBA $\mathcal{A}$, the value $\bigoplus_{t \in \Sigma^{K^*}} (\|\mathcal{A}\|, t)$ is computable in time polynomial in the size of $\mathcal{A}$.*

Proof. By combining the input alphabet Σ with the state set of $\mathcal{A}$, we can construct an automaton working over a singleton alphabet whose behavior on the unique input tree is exactly the desired supremum. We can then use the polynomial algorithm from [1] to compute this value. □

We will now look at the problem of deciding inclusion of the languages accepted by two automata and later show how this can be generalized to the weighted scenario.

3 Deciding Inclusion

We are interested in the inclusion problem of two automata, which can use different acceptance conditions. This problem is formally defined as follows.

Problem (Inclusion $\mathcal{I}_{\mathrm{X,Y}}$). Given an XA $\mathcal{A}$ and a YA $\mathcal{A}'$, decide whether $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$.

One approach to solving this problem is to construct an automaton that accepts the complement of $\mathcal{L}(\mathcal{A})$, since the inclusion $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$ holds iff $\mathcal{L}(\mathcal{A}') \cap \overline{\mathcal{L}(\mathcal{A})} = \emptyset$. If one is able to efficiently decide the emptiness of this intersection, then the inclusion problem can be easily solved. Thus, we look also at the complementation problem.

Problem (Complementation $\mathcal{C}_{\mathrm{X,Y}}$). Given an XA $\mathcal{A}$, construct a YA $\overline{\mathcal{A}}$ with $\mathcal{L}(\overline{\mathcal{A}}) = \overline{\mathcal{L}(\mathcal{A})}$.

Notice that we do not require that the complement automaton has the same acceptance condition as the original one. This is motivated by the fact that LA, BA and CA are not closed under complement [18, 14], but e.g., the complement of an LA is expressible through a BA (see Theorem 13).

Despite the difference in expressivity, the inclusion problem for Büchi automata $\mathcal{I}_{\mathrm{B,B}}$, for co-Büchi automata $\mathcal{I}_{\mathrm{C,C}}$ and for looping automata $\mathcal{I}_{\mathrm{L,L}}$ have the same complexity; they are all ExpTime-complete.

Theorem 11. *The problems $\mathcal{I}_{\mathrm{L,L}}$, $\mathcal{I}_{\mathrm{B,B}}$, and $\mathcal{I}_{\mathrm{C,C}}$ are* ExpTime*-complete.*

Proof. We show ExpTime-hardness of $\mathcal{I}_{\mathrm{L,L}}$ by reduction of the inclusion problem for finite trees, i.e., given two automata $\mathcal{A}$ and $\mathcal{A}'$ on finite trees, decide whether $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$. It was shown in [17, Theorem 2.1] that this problem is ExpTime-complete.

The reduction employs a straightforward translation of automata on finite trees to looping automata. Given an automaton $\mathcal{A} = (Q, \Sigma, I, \Delta)$ on finite trees, the equivalent LA $\mathcal{B} = (Q', \Sigma', I', \Delta')$ is constructed as follows:

- $Q' := Q \cup \{q_\star, q_0\}$, where $q_\star, q_0$ are new states.
- $\Sigma' := \Sigma \cup \{\star\}$, where $\star$ is a new symbol.
- $I' := I \cup \{q_0\}$.
- $\Delta' := \Delta \cup \{(q, \star, q_\star, \dots, q_\star) \mid q \in Q\} \cup \{(q_\star, \alpha', q_\star, \dots, q_\star) \mid \alpha' \in \Sigma'\} \cup \{(q_0, \alpha, q_1, \dots, q_k) \mid \exists 1 \leq i \leq k.q_i = q_0 \wedge \forall j \neq i.q_j = q_\star\}$.

In this construction every infinite tree $t' \in \Sigma'^{K^*}$ with a $\star$ on every path represents the finite tree $t \in \Sigma^T$ that is the subtree of t' before the first $\star$'s. $\mathcal{B}$ accepts all trees that have an infinite path without a $\star$ (guessed via q_0). On all trees that represent finite trees it behaves the same as $\mathcal{A}$. It is easy to see that $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$ holds for two automata on finite trees iff $\mathcal{L}(\mathcal{B}') \subseteq \mathcal{L}(\mathcal{B})$ holds for their corresponding looping automata.

We will now give an algorithm that decides $\mathcal{I}_{\mathrm{B,B}}$ in time exponential in the size of the input BA $\mathcal{A}$ and $\mathcal{A}'$. Let n and n' be the number of states of $\mathcal{A}$ and $\mathcal{A}'$, respectively.

1. We translate $\mathcal{A}$ into an equivalent APA $\mathcal{B}$. The transition function δ of this automaton can be determined as in Example 4. This construction yields an automaton with n states and 2 priorities.

2. We use [19, Lemma 6.8] to construct an equivalent PA $\mathcal{B}'$.[2] This non-deterministic automaton has a number of states exponential in n and a number of priorities polynomial in n. Let $2^{p(n)}$ be a bound on the number of states and $p'(n)$ be a bound on the number of priorities of $\mathcal{B}'$ for suitable polynomials p and p'.

3. Now we have to construct an automaton $\mathcal{C}$ recognizing the intersection of $\mathcal{L}(\mathcal{B}')$ and $\mathcal{L}(\mathcal{A}')$. To do this, we use a standard product construction on the automata, where the acceptance conditions have first been rewritten as Streett conditions. For $\mathcal{B}'$, the equivalent Streett condition has at most $p'(n)$ pairs and for $\mathcal{A}'$ we only need one pair. The product automaton then has as acceptance condition the conjunction of these two Streett conditions, which is again a Streett condition with at most $p'(n)+1$ pairs. The number of states of $\mathcal{C}$ is bounded by $n'2^{p(n)}$.

4. We rewrite the SA $\mathcal{C}$ again as a PA $\mathcal{C}'$. For this we use the construction in [4, Theorem 7]. This construction takes a finite-state Streett game and constructs an equivalent Rabin chain game. Unweighted automata can be interpreted as special finite-state games, so this result also holds for Streett automata and Rabin chain automata (see, e.g., [18]). Rabin chain conditions can equivalently be expressed as parity conditions of the same size.

   We arrive at a PA with $O(n'2^{p(n)}(p'(n)+1)!)$ states and $O(p'(n)+1)$ priorities. Thus, the number of states is bounded by $n'2^{r(n)}$ and the number of priorities by $r'(n)$ for polynomials r and r'.[3]

5. By testing emptiness of $\mathcal{L}(\mathcal{C}')$, we effectively decide the inclusion problem for $\mathcal{A}$ and $\mathcal{A}'$. It was shown in [12, Theorem 5.1 (1)] that emptiness of the parity automaton $\mathcal{C}'$ is decidable in time $O\left((n'2^{r(n)})^{r'(n)}\right)$, i.e., exponential in the number of states of $\mathcal{A}$.

This concludes the proof that $\mathcal{I}_{\mathrm{B,B}}$ is in ExpTime. The same procedure can be used to show that $\mathcal{I}_{\mathrm{C,C}}$ is in ExpTime, since co-Büchi automata are also easily expressed as parity or Streett automata. $\square$

The ExpTime-decision procedures for $\mathcal{I}_{\mathrm{B,B}}$ and $\mathcal{I}_{\mathrm{C,C}}$ provided by this proof are not practical, since they include several transformations into different automata

[2]In [19] alternating automata are defined differently, but the two descriptions can be transformed into each other in polynomial time.

[3]The factorial $x!$ is bounded by $x^x = 2^{x \log x} \leq 2^{x^2}$.

models. However, we can effectively give an exponential construction for the problem $\mathcal{C}_{\text{L,B}}$, which means that we can develop an optimal algorithm for solving $\mathcal{I}_{\text{L,B}}$.

Definition 12. Let $\mathcal{A} = (Q, \Sigma, I, \Delta)$ be an LA. The *complement automaton* of $\mathcal{A}$ is the BA $\overline{\mathcal{A}} := (Q_c, \Sigma, I_c, \Delta_c, F_c)$ where

- $Q_c := \mathcal{P}(Q)$.
- $I_c := \{I\}$.
- For $Q_0, \ldots, Q_k \subseteq Q$ and $\alpha \in \Sigma$, $(Q_0, \alpha, Q_1, \ldots, Q_k) \in \Delta_c$ iff
$$\bigforall_{q_0 \in Q_0} \; \bigforall_{y=(q_0,\alpha,q_1,\ldots,q_k)\in\Delta} \; \bigexists_{i\in K} q_i \in Q_i \ .$$
- $F_c := \{\emptyset\}$.

Notice that $\overline{\mathcal{A}}$ is exponential in the size of $\mathcal{A}$.

Theorem 13. *If $\mathcal{A}$ is an LA and $\overline{\mathcal{A}}$ its complement automaton, then $\mathcal{L}(\overline{\mathcal{A}}) = \overline{\mathcal{L}(\mathcal{A})}$.*

Proof. Let $t \in \mathcal{L}(\overline{\mathcal{A}})$. Then there is a successful run $\overline{r} \in Q_c^{K^*}$ of $\overline{\mathcal{A}}$ on t. Assume that there also is a valid run $r \in Q^{K^*}$ of $\mathcal{A}$ on t. We now inductively construct a path $p \in \mathcal{P}ath(K^*)$ for which $r(u) \in \overline{r}(u)$ holds for all nodes $u \in p$.

- For $u = \epsilon$ we have $r(\epsilon) \in I = \overline{r}(\epsilon)$.
- Let $u \in p$ be a node for which $r(u) \in \overline{r}(u)$ holds. Since r and $\overline{r}$ are valid, we have $(r(u), t(u), r(u1), \ldots, r(uk)) \in \Delta$ and $(\overline{r}(u), t(u), \overline{r}(u1), \ldots, \overline{r}(uk)) \in \Delta_c$. By definition of Δ_c, there must be an $i \in K$ with $r(ui) \in \overline{r}(ui)$. We now append ui to the path p and continue.

The run $\overline{r}$ cannot fulfill the final state condition $\{\emptyset\}$ of $\overline{\mathcal{A}}$ on the path p, since every label along the path must contain at least one element. This contradicts the fact that $\overline{r}$ is successful, and thus t cannot be accepted by $\mathcal{A}$.

For the other inclusion, let $t \notin \mathcal{L}(\mathcal{A})$. By Corollary 8, there must be a closed, finite subtree $T \subseteq K^*$ on which no valid subrun exists. We now inductively construct a successful run $\overline{r} \in Q_c^{K^*}$ of $\overline{\mathcal{A}}$ on t for which every node $u \in T$ has the following property:

$$P(u) \quad \equiv \quad \bigforall_{r \in Q^{u[K^*]}} \left[r(u) \in \overline{r}(u) \to \left(\bigexists_{w \in u[K^*] \cap \text{int}(T)} \overrightarrow{r(t,w)} \notin \Delta \right) \right]$$

This means that every mapping $r \in Q^{u[K^*]}$ that starts in a state $q_0 \in \overline{r}(u)$ at u must violate Δ at some node in $\text{int}(T)$ that lies below u.

- If we set $\overline{r}(\epsilon) := \{I\}$, then $P(\epsilon)$ holds because of Corollary 8.
- If u is a leaf of T or $u \notin T$, we set $\overline{r}(ui) := \emptyset$ for each $i \in K$.
- Let now u be an inner node of T where $\overline{r}(u)$ has already been defined and $P(u)$ holds. We initially set $\overline{r}(ui) := \emptyset$ for every $i \in K$. Thus, $P(ui)$ trivially holds for every $i \in K$, but the transition $\overrightarrow{\overline{r}(t,u)}$ need not be valid. We now have to expand the label sets $\overline{r}(ui)$ in such a way that

  1. the transition $\overrightarrow{\overline{r}(t,u)}$ becomes valid and
  2. the properties $P(ui)$ are not violated.

  We do this by checking the conditions of Δ_c step by step.

  – Let $q_0 \in \overline{r}(u)$ and $y = (q_0, t(u), q_1, \ldots, q_k) \in \Delta$.

  – Assume that for each index $i \in K$ there is a mapping $r_i \in Q^{ui[K^*]}$ with $r_i(ui) = q_i$ that does not violate Δ below ui in $\mathrm{int}(T)$. Then we could join these mappings into a mapping $r \in Q^{u[K^*]}$ with $r(u) := q_0$ and $r(uiw) := r_i(uiw)$ for all $i \in K$ and $w \in K^*$. This mapping does not violate Δ below u in $\mathrm{int}(T)$, which contradicts $P(u)$.

  – Thus we can find an index $i \in K$ such that $P(ui)$ still holds after we add q_i to $\overline{r}(ui)$.

  After we have done this for every $q_0 \in \overline{r}(u)$ and every matching transition $y \in \Delta$, we have fully determined the successor labels $\overline{r}(ui)$ and $P(ui)$ still holds for every $i \in K$. Additionally, $\overrightarrow{\overline{r}(t,u)}$ now is a valid transition in Δ_c.

To show that $\overline{r}$ is a valid run of $\overline{\mathcal{A}}$ on t, we need to show that every transition is compatible with Δ_c. If the transition fully lies in T or $\overline{T}$, this is clear from the construction.

Let now $u \in \mathrm{fr}(T)$. Since $P(u)$ holds, all mappings $r \in Q^{u[K^*]}$ with $r(u) \in \overline{r}(u)$ must violate Δ in $u[K^*] \cap \mathrm{int}(T) = \emptyset$, which is clearly not possible. This implies that $\overline{r}(u) = \emptyset$, and thus, the transition $\overrightarrow{\overline{r}(t,u)} = (\emptyset, t(u), \emptyset, \ldots, \emptyset)$ is valid in Δ_c.

It is clear that $\overline{r}$ is successful since every infinite path must leave T at some node u and thus has the label $\emptyset$ at every node below u. This implies $t \in \mathcal{L}(\overline{\mathcal{A}})$. □

We can in fact improve this result by giving a construction that solves $\mathcal{C}_{\mathrm{C,GB}}$. Since every GBA can be transformed into a BA in polynomial time, this gives us a solution also for $\mathcal{C}_{\mathrm{C,B}}$, and hence for $\mathcal{I}_{\mathrm{C,B}}$. The following construction follows the ideas developed in [15] for simulating alternating tree automata using nondeterministic automata.

Definition 14. The *complement automaton* of a CA $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ is the GBA $\overline{\mathcal{A}} := (Q_c, \Sigma, I_c, \Delta_c, F_{c,1}, \ldots, F_{c,|F|+1})$, where

- Q_c contains all tuples of the form $(Q_0, Q_1, \dots, Q_{|F|+1})$, where $Q_0 \subseteq Q \setminus F$ and $Q_1, \dots, Q_{|F|+1} \subseteq F$ are disjoint.
- $I_c := \{(I \setminus F, I \cap F, \emptyset, \dots, \emptyset)\}$.
- From a state $(Q_0^{(0)}, Q_1^{(0)}, \dots, Q_{|F|+1}^{(0)})$ at a node labeled with α, the transition to $((Q_0^{(1)}, Q_1^{(1)}, \dots, Q_{|F|+1}^{(1)}), \dots, (Q_0^{(k)}, Q_1^{(k)}, \dots, Q_{|F|+1}^{(k)}))$ is allowed iff
$$\bigforall_{j=0}^{|F|+1} \bigforall_{q \in Q_j^{(0)}} \bigforall_{(q,\alpha,q_1,\dots,q_k) \in \Delta} \left(\bigexists_{\substack{i \in K \\ q_i \in F}} \bigexists_{l=\max\{j,1\}}^{|F|+1} q_i \in Q_l^{(i)} \right) \vee \left(\bigexists_{\substack{i \in K \\ q_i \notin F}} q_i \in Q_0^{(i)} \right) .$$
- $F_{c,j} := \{(Q_0, Q_1, \dots, Q_{|F|+1}) \in Q_c \mid Q_j = \emptyset\}$ for $j \in \{1, \dots, |F|+1\}$.

The proof that this construction is correct closely follows the main ideas used in the proof of Theorem 13. We will now show the correctness in two steps.

Lemma 15. *Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a CA and $\overline{\mathcal{A}}$ its complement automaton. Then $\mathcal{L}(\overline{\mathcal{A}}) \subseteq \overline{\mathcal{L}(\mathcal{A})}$.*

Proof. Let $t \in \mathcal{L}(\overline{\mathcal{A}})$, i.e., there is a successful run $\overline{r} \in Q_c^{K^*}$ of $\overline{\mathcal{A}}$ on t, and assume that there also is a successful run $r \in Q^{K^*}$ of $\mathcal{A}$ on t. For a node $u \in K^*$, define $\overline{R}(u) := \bigcup_{j=0}^{|F|+1} \overline{r}(u)_j$. Then we can inductively construct an infinite path $p \in \mathcal{P}ath(K^*)$ for which $r(u) \in \overline{R}(u)$ holds for all $u \in p$:

- Since $r(\varepsilon) \in I$, either $r(\varepsilon) \in I \setminus F = \overline{r}(u)_0$ or $r(\varepsilon) \in I \cap F = \overline{r}(u)_1$ must hold, and thus $r(\varepsilon) \in \overline{R}(\varepsilon)$.
- Let $u \in p$ be a node with the property $r(u) \in \overline{R}(u)$. Since $\overrightarrow{r(t,u)} \in \Delta$ and $\overrightarrow{\overline{r}(t,u)} \in \Delta_c$, there must be an $i \in K$ such that $r(ui) \in \overline{r}(ui)_j$ holds for some $j \in \{0, \dots, |F|+1\}$. Thus $r(ui) \in \overline{R}(ui)$ holds and we can append ui to the path p.

Since r is successful, there must be a node $u_0 \in p$ such that $r(u) \in F$ holds for all nodes $u \in p \cap u_0[K^*]$ that occur below u_0 along the path p. That means that $r(u)$ always occurs in a component $\overline{r}(u)_j$ with $j \geq 1$. The index j of this component can only grow bigger or stay the same with each transition, and thus there must be a node $u_1 \in p$ after which $r(u) \in \overline{r}(u)_j$ always holds for some fixed $j \in \{1, \dots, |F|+1\}$. Thus $\overline{r}(u)_j$ can never be empty after the node u_1 along the path p, which contradicts the success of $\overline{r}$. □

For this direction, it is easy to see the similarity to the proof of Theorem 13. The other direction is also similar. The property $P(u)$ is replaced by a more complex property **Fail**(u) and the proof is generally more complex to account for the different components of each state. Instead of Corollary 8, we have to use the more general version in Corollary 7 for this proof.

Lemma 16. *Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a CA and $\overline{\mathcal{A}}$ its complement automaton. Then $\mathcal{L}(\overline{\mathcal{A}}) \supseteq \overline{\mathcal{L}(\mathcal{A})}$.*

Proof. Let $t \notin \mathcal{L}(\mathcal{A})$. We inductively construct a successful run $\overline{r} \in Q_c^{K^*}$ of $\overline{\mathcal{A}}$ on t. For every node $u \in K^*$ the following property $\mathbf{Fail}(u)$ will be satisfied.

$$\mathbf{Fail}(u) \equiv \bigforall_{j=0}^{|F|+1} \bigforall_{r \in Q^{u[K^*]}} r(u) \in \overline{r}(u)_j \to \bigexists_{w \in u[K^*]} \mathbf{Fail}(w, \overrightarrow{r(t,w)})$$

$$\mathbf{Fail}(u, y = (q_0, \ldots)) \equiv$$

$$y \in \Delta \to \left(q_0 \notin F \wedge \bigforall_{\substack{r' \in Q^{u[K^*]} \\ r'(u) = q_0}} \neg\mathbf{Valid}(r', u) \vee \neg\mathbf{Success}(r', u) \right)$$

$$\mathbf{Valid}(r, u) \equiv \bigforall_{w \in u[K^*]} \overrightarrow{r(t,w)} \in \Delta$$

$$\mathbf{Success}(r, u) \equiv \bigforall_{p \in \mathcal{P}ath(u[K^*])} \mathrm{Inf}(r, p) \setminus F = \emptyset$$

$\mathbf{Success}(r, u)$ expresses that a run r is "successful below u", i.e., all infinite paths starting from u must contain only finitely many states from $Q \setminus F$.[4] The property $\mathbf{Valid}(r, u)$ ensures that all transitions of a run r below a node u are valid transitions of $\mathcal{A}$. Using these two properties, we formulate $\mathbf{Fail}(u, y)$ by saying that if y is a valid transition at u, then the current state must be non-final and no valid run starting from this state can be successful. Finally, $\mathbf{Fail}(u)$ says that every run starting in a state occurring in $\overline{r}(u)$ must fail somewhere below u.

The property $\mathbf{Fail}(u, y)$ is clearly of the form required by Corollary 7, and thus $\mathbf{Fail}(u)$ is equivalent to a property $\mathbf{Fail}(u, (T_{j,u}))$ for closed, finite trees $T_{j,u} \subseteq u[K^*]$ ($j \in \{1, \ldots, |F| + 1\}$). This property is the same as $\mathbf{Fail}(u)$, except that "$w \in u[K^*]$" is replaced by "$w \in T_{j,u}$".[5]

To start the construction of $\overline{r}$, we set $\overline{r}(\varepsilon) := (I \setminus F, I \cap F, \emptyset, \ldots, \emptyset)$ and deduce $\mathbf{Fail}(\varepsilon)$ as follows. If $\mathbf{Fail}(\varepsilon)$ was not fulfilled, there would be a run $r \in Q^{K^*}$ with $r(\varepsilon) \in I$ for which all transitions are valid and for every $w \in K^*$ with $r(w) \notin F$ there would be a run $r'_w \in Q^{w[K^*]}$ with $r'_w(w) = r(w)$ that is both valid and successful below w. Then we could construct a run $r' \in Q^{K^*}$ by replacing the

[4] $\mathrm{Inf}(r, p)$ denotes the set of states occurring infinitely often in r along p.

[5] The tree $T_{0,u}$ can always be chosen to be the singleton tree $\{u\}$: If every valid run starting in a state from $Q \setminus F$ at u must contain a node w with $\mathbf{Fail}(w, \overrightarrow{r(t,w)})$, then for every such run $\mathbf{Fail}(u, \overrightarrow{r(t,u)})$ will already be satisfied. This is because any path containing w must also contain u.

labels of r on the subtree $w[K^*]$ with those of r'_w at every such node $w \in K^*$.[6] This run r' would be a valid and successful run of $\mathcal{A}$ on t, which contradicts the assumption $t \notin \mathcal{L}(\mathcal{A})$.

Suppose now that $u \in K^*$ is a node where $\overline{r}(u)$ has already been defined and for which $\mathbf{Fail}(u, (T_{j,u}))$ holds for some finite trees $T_{j,u} \subseteq u[K^*]$ $(j \in \{1, \dots, |F|+1\})$. For every $i \in K$ we construct $\overline{r}(ui)$ from $\overline{r}(u)$ in several steps.

- First we determine an index $\widetilde{j} \in \{1, \dots, |F|+1\}$ with $\overline{r}(u)_{\widetilde{j}} = \emptyset$. Since we will keep the sets $\overline{r}(u)_j$ $(j \in \{1, \dots, |F|+1\})$ disjoint, there can be at most $|F|$ non-empty sets and thus such an index $\widetilde{j}$ can always be chosen.
- We initially set $\overline{r}(ui) := (\emptyset, \dots, \emptyset)$ for each $i \in K$, and thus $\mathbf{Fail}(ui)$ holds for our initial definiton of $\overline{r}(ui)$. But clearly, the resulting transition $\overrightarrow{\overline{r}(t,u)}$ need not satisfy the transition relation Δ_c. We now enlarge the sets $\overline{r}(ui)$ in such a way that $\mathbf{Fail}(ui)$ remains satisfied and $\overrightarrow{\overline{r}(t,u)}$ becomes a valid transition.
- For every $j \in \{0, \dots, |F|+1\}$, $q \in \overline{r}(u)_j$ and $y = (q, t(u), q_1, \dots, q_k) \in \Delta$, we do the following.
  - We choose one index $i \in K$ for which q_i is added to a component of $\overline{r}(ui)$. The index of this new component is determined as follows:
    - If $q_i \notin F$, we set $\overline{r}(ui)_0 := \overline{r}(ui)_0 \cup \{q_i\}$.
    - If $j > 0$ and $q_i \in F$, we set $\overline{r}(ui)_j := \overline{r}(ui)_j \cup \{q_i\}$.
    - If $j = 0$ and $q_i \in F$, we set $\overline{r}(ui)_{\widetilde{j}} := \overline{r}(ui)_{\widetilde{j}} \cup \{q_i\}$.

    We choose i such that $\mathbf{Fail}(ui)$ remains satisfied after we add q_i to $\overline{r}(ui)$ as specified above. As we will show in the following, such an index always exists. For this, we make a case distinction depending on whether $q \in F$ or not.
    - Let $q \notin F$, i.e., $j = 0$ and assume that $\mathbf{Fail}(ui)$ is violated by adding q_i to $\overline{r}(ui)$. Then there are subruns $r_i \in Q^{ui[K^*]}$ with the properties
      - $r_i(ui) = q_i$ and
      - $\mathbf{Fail}(w, \overrightarrow{r_i(t,w)})$ is not satisfied for any $w \in ui[K^*]$, i.e., if $w \notin F$, then there is a valid and successful subrun $r'_w \in Q^{w[K^*]}$ with $r'_w(w) = r_i(w)$.

      As in the argument for $\mathbf{Fail}(\varepsilon)$, we can now construct a subrun $r' \in Q^{u[K^*]}$ with $\overrightarrow{r'(t,u)} = y$ which is valid and successful. This means that $\mathbf{Fail}(u, y)$ is not satisfied. If we now construct the subrun

[6] We only do this replacement for the first occurrence of a state from $Q \setminus F$, not in a subtree that has already been replaced.

$r \in Q^{u[K^*]}$ by concatenating y and the subruns r_i, $\mathbf{Fail}(w, \overrightarrow{r(t,w)})$ is not satisfied for any $w \in u[K^*]$, which is a contradiction to $\mathbf{Fail}(u)$.

    * If $q \in F$, i.e., $j \geq 1$, we could use the same argument as above. However, in this case we take a closer look at the finite tree $T_{j,ui}$ because this will later enable us to show that $\overline{r}$ is successful. Since all q_i are added to either $\overline{r}(ui)_0$ or $\overline{r}(ui)_j$, we need only be concerned with the trees $T_{0,ui}$ and $T_{j,ui}$. We will show that we can choose $i \in K$ such that the property $\mathbf{Fail}(ui, (T_{j',ui}))$ remains satisfied if we set $T_{0,ui} := T_{j,ui} := T_{j,u} \cap ui[K^*]$.
      If we assume the converse, we could deduce that there exist subruns $r_i \in Q^{ui[K^*]}$ with the following properties:
      · $r_i(ui) = q_i$.
      · $\mathbf{Fail}(w, \overrightarrow{r_i(t,w)})$ is not satisfied for any $w \in T_{j,u} \cap ui[K^*]$.
      If we now construct the subrun $r \in Q^{u[K^*]}$ by concatenating the transition y and the subruns r_i, then it is easily seen that r starts in $r(u) = q$ and no $\mathbf{Fail}(w, \overrightarrow{r(t,w)})$ is satisfied for any $w \in T_{j,u} \cap ui[K^*]$ and for any $i \in K$. Furthermore, $\mathbf{Fail}(u, \overrightarrow{r(t,u)})$ is also not satisfied, since $\overrightarrow{r(t,u)} = y \in \Delta$, but $q \in F$. This means that r is a counterexample to $\mathbf{Fail}(u, (T_{j',u}))$.

  - After we have done this for every j, q and y, the transition $\overrightarrow{\overline{r}(t,u)}$ is valid and the properties $\mathbf{Fail}(ui)$ still hold.

- As a last step, we need to make sure that the sets $\overline{r}(ui)_1, \ldots, \overline{r}(ui)_{|F|+1}$ are disjoint for every $i \in K$. To do this, we remove all but the rightmost occurrence of each state $q \in F$ in these sets. The transition $\overrightarrow{\overline{r}(t,u)}$ remains valid, because Δ_c only requires a state q_i to be present in some position l that is greater than or equal to $\max\{j, 1\}$. The properties $\mathbf{Fail}(ui)$ also still hold, because we only *removed* states from some of the components of $\overline{r}(ui)$.

- Since $\mathbf{Fail}(ui)$ holds, there are finite trees $T_{j,ui}$ ($j \in \{1, \ldots, |F|+1\}$) such that $\mathbf{Fail}(ui, (T_{j,ui}))$ holds. These trees can be determined as follows.

  - $T_{0,ui}$ can be set to $\{ui\}$ since $\mathbf{Fail}(ui)$ implies that for any run $r \in Q^{ui[K^*]}$ with $r(ui) \in \overline{r}(ui)_0$ the property $\mathbf{Fail}(ui, \overrightarrow{r(t,ui)})$ must hold.
  - $T_{\widetilde{j},ui}$ must be determined from $\mathbf{Fail}(ui)$ using Corollary 7.
  - For any j that is not 0 or $\widetilde{j}$, we can set $T_{j,ui} := T_{j,u} \cap ui[K^*]$. This is possible because of the way we constructed $\overline{r}(ui)_j$.

It remains to show that $\overline{r}$ is a successful run of $\overline{\mathcal{A}}$. For this we assume that there is a path $p \in \mathcal{P}ath(K^*)$ such that for some $j \in \{1, \ldots, |F|+1\}$ the set $\overline{r}(u)_j$ is empty only finitely often for nodes $u \in p$. Then there is a node $u \in p$ after which no empty set occurs in the j-th component of $\overline{r}$ along p. By construction of $\overline{r}$, the property $\mathbf{Fail}(u, (T_{j',u}))$ must be satisfied for finite trees $T_{j',u} \subseteq u[K^*]$.

Let v be the first node of p that lies outside of $T_{j,u}$. By construction of $\overline{r}$, $\mathbf{Fail}(v, (T_{j',v}))$ must hold for finite trees $T_{j',v} \subseteq v[K^*]$. The tree $T_{j,v}$ can be chosen to be $T_{j,u} \cap v[K^*] = \emptyset$ since no empty set occurred in the j-th component along the path from u to v. Since $\mathbf{Fail}(v, (T_{j',v}))$ is satisfied, this means that $\overline{r}(v)$ must be empty, which contradicts the assumption. Thus, $\overline{r}$ is a successful run of $\overline{\mathcal{A}}$ on t and $t \in \mathcal{L}(\overline{\mathcal{A}})$. □

We obtain the following theorem.

Theorem 17. *If $\mathcal{A}$ is a CA and $\overline{\mathcal{A}}$ its complement automaton, then $\mathcal{L}(\overline{\mathcal{A}}) = \overline{\mathcal{L}(\mathcal{A})}$.* □

Notice that the complement automaton of a CA $\mathcal{A}$ is exponential in the size of $\mathcal{A}$. This implies that $\mathcal{C}_{\mathrm{C,B}}$ (and thus, also $\mathcal{I}_{\mathrm{C,B}}$) is solvable in exponential time. We hence obtain the following corollary.

Corollary 18. *The inclusion problem $\mathcal{I}_{\mathrm{C,B}}$ is* ExpTime*-complete.*

Proof. Since LA are a special kind of BA and CA, ExpTime-hardness follows from the hardness of $\mathcal{I}_{\mathrm{L,L}}$ (see Theorem 11). The upper bound is a direct consequence of Theorem 17 and Lemmata 9 and 10. □

Unfortunately, a similar construction for the problem $\mathcal{C}_{\mathrm{B,C}}$ is not possible. This was shown in [13] by means of a counterexample, i.e., a tree language that is recognizable by a BA, but whose complement is not recognizable by a CA.

4 The Weighted Inclusion Problem

As mentioned already, unweighted automata can be seen as weighted automata over a very simple Boolean lattice, namely $\mathbb{B}$, whose operators correspond to the logical connectives. In fact, Boolean lattices can be seen as the natural generalization of Boolean logic, where the conjunction, disjunction and negation are translated to the infimum $\otimes$, supremum $\oplus$, and complementation $^-$. We can use this fact to describe "natural" generalizations of the decision problems for unweighted automata to the weighted setting.

From a low-level point of view, the inclusion problem consists of deciding whether the implication $t \in \mathcal{L}(\mathcal{A}') \Rightarrow t \in \mathcal{L}(\mathcal{A})$ holds for every input tree t. Equivalently, we can express this property using the formula:

$$\bigwedge_{t \in \Sigma^{K^*}} \neg(\|\mathcal{A}'\|, t) \vee (\|\mathcal{A}\|, t) \ ,$$

which can then be generalized to arbitrary Boolean lattices as follows.

Problem (Weighted Inclusion $\mathcal{I}_{\mathrm{WX,WY}}$). Given a WXA $\mathcal{A}$ and a WYA $\mathcal{A}'$ over the same Boolean lattice, compute $\bigotimes_{t\in\Sigma^{K^*}} \overline{(\|\mathcal{A}'\|, t)} \oplus (\|\mathcal{A}\|, t)$.

Remark. A more intuitive generalization of the inclusion problem is to decide whether $(\|\mathcal{A}'\|, t) \leq (\|\mathcal{A}\|, t)$ holds for all input trees t. This is, however, only a special case of the above problem, since

$$(\|\mathcal{A}'\|, t) \leq (\|\mathcal{A}\|, t) \Leftrightarrow \overline{(\|\mathcal{A}'\|, t)} \oplus (\|\mathcal{A}\|, t) = 1_S \ .$$

Related to inclusion is the problem of deciding the equivalence of two unweighted automata, which can be decided by two inclusion tests. Generalizing this to Boolean lattices, one can compute the *weighted equivalence* of a WXA $\mathcal{A}$ and a WYA $\mathcal{A}'$ as the infimum of the two weighted inclusions of type $\mathcal{I}_{WX,WY}$ and $\mathcal{I}_{WY,WX}$. This value expresses the degree to which the two automata recognize the same tree series and will be 1_S iff these tree series are equal.

As in the unweighted case, the problem $\mathcal{I}_{\mathrm{WX,WY}}$ can sometimes be reduced to a (lattice) complementation problem.

Problem (Weighted Complementation $\mathcal{C}_{\mathrm{WX,WY}}$). Given a WXA $\mathcal{A}$, construct a WYA $\overline{\mathcal{A}}$ such that $(\|\overline{\mathcal{A}}\|, t) = \overline{(\|\mathcal{A}\|, t)}$ holds for every $t \in \Sigma^{K^*}$.

Similar to the unweighted case, this reduction is based on the feasibility of computing the behavior of the binary infimum of two automata. This task is of polynomial complexity for weighted Büchi tree automata (see Lemmata 9 and 10).

We now present two methods for solving the weighted inclusion problem over arbitrary Boolean lattices. The first method uses the algorithm for testing the inclusion of unweighted automata as a black-box. This algorithm is called several times in a systematic way until the desired aggregated infimum is found.

The second method uses a glass-box approach instead, i.e., it modifies the original inclusion algorithms from the previous section to perform the computations over the lattice directly, without the need of repeatedly testing for inclusion. Our transformation uses a straightforward translation of the logical operators into their lattice counterparts. A surprising result is that the black-box approach is in fact more efficient than the glass-box.

4.1 Black-Box Approach

Since we already have a decision procedure for the unweighted problem $\mathcal{I}_{\mathrm{B,B}}$, we can use this to construct a *black-box* algorithm for $\mathcal{I}_{\mathrm{WB,WB}}$. This approach reduces $\mathcal{I}_{\mathrm{WB,WB}}$ to several inclusion checks. The main advantage of such an approach is that one can use any procedure deciding the unweighted problem, including any optimizations developed for it, since this procedure needs not be modified.

The black-box reduction of $\mathcal{I}_{\mathrm{WB,WB}}$ to $\mathcal{I}_{\mathrm{B,B}}$ is based on an idea from [6] and exploits the fact that every lattice element can be represented as the infimum of all the meet prime elements above it.

Let $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt}, F)$ and $\mathcal{A}' = (Q', \Sigma, S, \mathrm{in}', \mathrm{wt}', F')$ be two WBA over the same lattice S and $p \in S$ a meet prime element. We define the *cropped automata* $\mathcal{A}_p$ and $\mathcal{A}'_{\overline{p}}$ as the BA $(Q, \Sigma, I, \Delta, F)$ and $(Q', \Sigma, I', \Delta', F')$, respectively, where the initial state sets and transition relations are as follows:

- $I := \{q \in Q \mid \mathrm{in}(q) \not\leq p\}$, $\Delta := \{y \in Q \times \Sigma \times Q^k \mid \mathrm{wt}(y) \not\leq p\}$,
- $I' := \{q' \in Q' \mid \mathrm{in}'(q') \geq \overline{p}\}$, $\Delta' := \{y' \in Q' \times \Sigma \times Q'^k \mid \mathrm{wt}'(y') \geq \overline{p}\}$.

The transitions allowed in $\mathcal{A}_p$ ($\mathcal{A}'_{\overline{p}}$) are exactly those transitions having weight $\not\leq p$ ($\geq \overline{p}$) in $\mathcal{A}$ ($\mathcal{A}'$). It is easy to show that this property is transferred to the behavior of the weighted automata as follows. We have $(\|\mathcal{A}\|, t) \leq p$ iff $t \notin \mathcal{L}(\mathcal{A}_p)$ and $(\|\mathcal{A}'\|, t) \geq \overline{p}$ iff $t \in \mathcal{L}(\mathcal{A}'_{\overline{p}})$ for all $t \in \Sigma^{K^*}$. From this, it follows that $\bigotimes_{t \in \Sigma^{K^*}} (\|\mathcal{A}\|, t) \oplus \overline{(\|\mathcal{A}'\|, t)} \leq p$ holds iff $\mathcal{L}(\mathcal{A}'_{\overline{p}}) \not\subseteq \mathcal{L}(\mathcal{A}_p)$.

We have assumed that the Boolean lattice S is generated by the elements in the images of the initial distribution and transition weight functions of $\mathcal{A}$ and $\mathcal{A}'$. Since the number of meet prime elements in any distributive lattice is at most exponential in the number of elements generating it,[7] S has at most exponentially many meet prime elements measured on the sizes of $\mathcal{A}$ and $\mathcal{A}'$. Thus, this black-box approach requires at most exponentially many inclusion tests, each of which is itself exponential in the sizes of these automata. This means that $\mathcal{I}_{\mathrm{WB,WB}}$ is solvable in exponential time.

Notice, additionally, that the reduction we used depends only on the number of meet prime elements and on the existence of an exponential-time inclusion test for the unweighted version of the automata, but not on the specific acceptance condition used. In other words, if $\mathcal{I}_{\mathrm{X,Y}}$ can be decided in exponential time, then $\mathcal{I}_{\mathrm{WX,WY}}$ is computable in exponential time, too.

Theorem 19. *Let S be a Boolean lattice and X, Y be two acceptance conditions such that the problem $\mathcal{I}_{\mathrm{X,Y}}$ is decidable in some complexity class* C *that includes* ExpTime. *Then the problem of deciding whether a given value $a \in S$ solves an instance of $\mathcal{I}_{\mathrm{WX,WY}}$ is also in* C. □

Corollary 20. *Let S be a Boolean lattice. The problems of deciding whether a given value $a \in S$ is the solution of $\mathcal{I}_{\mathrm{WB,WB}}$, $\mathcal{I}_{\mathrm{WL,WL}}$, $\mathcal{I}_{\mathrm{WC,WC}}$, $\mathcal{I}_{\mathrm{WL,WB}}$, or $\mathcal{I}_{\mathrm{WC,WB}}$ are* ExpTime*-complete.* □

[7] Each meet prime element can be expressed as the supremum of some generating elements and complements of generating elements.

4.2 Glass-Box Approach

We now describe a construction that computes $\mathcal{I}_{\mathrm{WX,WY}}$ directly, rather than by means of several inclusion tests. This construction is in fact a generalization of the method used for deciding inclusion of unweighted automata presented in the previous section; hence the name *glass-box*.

Recall from Section 3 that the procedure deciding inclusion of two automata $\mathcal{A}$ and $\mathcal{A}'$ (i.e., whether $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$) required three steps: first construct an automaton $\overline{\mathcal{A}}$ accepting the complement of $\mathcal{L}(\mathcal{A}')$; then, intersect $\mathcal{A}'$ and $\mathcal{A}$, and finally decide emptiness of the resulting automaton. We have shown that the last two steps can be solved for weighted Büchi automata in polynomial time (see Lemmata 9 and 10). Thus, if we can solve the problem $\mathcal{C}_{\mathrm{WX,WB}}$, then we will also have a procedure that solves $\mathcal{I}_{\mathrm{WX,WB}}$.

Let us consider first the case of complementing looping automata. Definition 12 shows us how to build an automaton that accepts the complement language of a given looping automaton. Notice first that the transition relation of the automaton $\overline{\mathcal{A}}$ is equivalent to the following formula:

$$\bigwedge_{(q_0,\alpha,q_1,\ldots,q_k)\in Q\times\Sigma\times Q^k} q_0 \notin Q_0 \vee y \notin \Delta \vee \bigvee_{i\in K} q_i \in Q_i \ .$$

If we see this construction as a weighted automaton over the lattice $\mathbb{B}$, then it is easy to see how to generalize it to arbitrary Boolean lattices.

Definition 21. The *complement automaton* of a WLA $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt})$ is the WBA $\overline{\mathcal{A}} = (Q_c, \Sigma, S, \mathrm{in}_c, \mathrm{wt}_c, F_c)$ where

- $Q_c := S^Q$.
- For $\varphi \in Q_c$, $\mathrm{in}_c(\varphi) := \begin{cases} 1_S & \text{if } \varphi(q) \geq \mathrm{in}(q) \text{ for all } q \in Q \\ 0_S & \text{otherwise} \end{cases}$.
- For $\varphi_0, \ldots, \varphi_k \in Q_c$ and $\alpha \in \Sigma$, $\mathrm{wt}_c(\varphi_0, \alpha, \varphi_1, \ldots, \varphi_k) :=$

$$\bigotimes_{y=(q_0,\alpha,q_1,\ldots,q_k)\in Q\times\{\alpha\}\times Q^k} \overline{\varphi_0(q_0)} \oplus \overline{\mathrm{wt}(y)} \oplus \bigoplus_{i\in K} \varphi_i(q_i) \ .$$

- $F_c := \{\underline{0_S}\}$ where $\underline{0_S} : Q \to S : q \mapsto 0_S$.

We now show that this construction solves the weighted complementation problem $\mathcal{C}_{\mathrm{WL,WB}}$. For this, we fix a WLA $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt})$ and an input tree $t \in \Sigma^{K^*}$ and need to show that $(\|\overline{\mathcal{A}}\|, t) = \overline{(\|\mathcal{A}\|, t)}$ holds. The next two sections are dedicated to proving the two halves of this claim. The proof uses similar ideas to those of Theorem 13, generalized to Boolean lattices.

4.2.1 Proof of $(\|\overline{\mathcal{A}}\|, t) \leq \overline{(\|\mathcal{A}\|, t)}$

We show this direction by proving the inequality $\mathrm{wt}_c(t, \overline{r}) \leq \overline{\mathrm{wt}(t, r)}$ for all $\overline{r} \in \mathrm{succ}(\overline{\mathcal{A}})$ and $r \in Q^{K^*}$. If $\mathrm{wt}_c(t, \overline{r}) = 0_S$ or $\mathrm{wt}(t, r) = 0_S$ this is trivially satisfied, so we fix two runs $\overline{r} \in \mathrm{succ}(\overline{\mathcal{A}})$ and $r \in Q^{K^*}$ with $\mathrm{wt}_c(t, \overline{r}) > 0_S$ and $\mathrm{wt}(t, r) > 0_S$.

We proceed by showing that $\mathrm{wt}_c(t, \overline{r}) \otimes \mathrm{wt}(t, r)$ is smaller than $a \otimes \overline{a} = 0_S$ for some suitably chosen $a \in S$. Looking at Theorem 13 one can already guess that this argument has to do with paths $p \in \mathcal{P}ath(K^*)$ for which $r(u) \in \overline{r}(u)$ holds for all $u \in p$. In the weighted case, this property is replaced by the value $\bigotimes_{u \in p} \overline{r}(u)(r(u))$. To be exact, a has the form

$$\bigoplus_{p \in \mathcal{P}ath(K^*, n)} \bigotimes_{u \in p} \overline{r}(u)(r(u))$$

for some $n \in \mathbb{N}$.

Lemma 22. *There is a depth $m \in \mathbb{N}$ such that*

$$\mathrm{wt}_c(t, \overline{r}) \leq \bigotimes_{p \in \mathcal{P}ath(K^*, m)} \bigoplus_{u \in p} \overline{\overline{r}(u)(r(u))} \ .$$

Proof. Since $\overline{r}$ is successful, there is a minimal depth $m \in \mathbb{N}$ such that any path p visits at least one node labeled by $\underline{0_S}$ before reaching depth m.

Let now $p \in \mathcal{P}ath(K^*, m)$ and assume that $\mathrm{wt}_c(t, \overline{r}) \not\leq \bigoplus_{u \in p} \overline{\overline{r}(u)(r(u))}$. Then $\bigoplus_{u \in p} \overline{\overline{r}(u)(r(u))} < 1_S$ and thus $\overline{r}(u)(r(u)) > 0_S$ holds for every $u \in p$. Hence there cannot be a node labeled with $\underline{0_S}$ along p in $\overline{r}$, which contradicts the above choice of m. $\square$

We now show the second part, which leads to the "contradiction" $\mathrm{wt}(t, r) \otimes \mathrm{wt}_c(t, \overline{r}) \leq 0_S$.

Lemma 23. *For all $n \in \mathbb{N}$ the following inequation holds:*

$$\mathrm{wt}(t, r) \otimes \mathrm{wt}_c(t, \overline{r}) \leq \bigoplus_{p \in \mathcal{P}ath(K^*, n)} \bigotimes_{u \in p} \overline{r}(u)(r(u)) \ .$$

Proof. For $n = 0$ we have

$$\mathrm{wt}(t, r) \otimes \mathrm{wt}_c(t, \overline{r}) \leq \mathrm{in}(r(\epsilon)) \leq \overline{r}(\epsilon)(r(\epsilon)) = \bigoplus_{p \in \mathcal{P}ath(K^*, 0)} \bigotimes_{u \in p} \overline{r}(u)(r(u)) \ .$$

This holds since $\mathrm{wt}_c(t, \overline{r}) > 0_S$ and thus $\mathrm{in}_c(\overline{r}(\epsilon)) > 0_S$ and $\overline{r}(\epsilon)(r(\epsilon)) \geq \mathrm{in}(r(\epsilon))$.

Let now the inequation hold for some $n \in \mathbb{N}$. For $p \in \mathcal{P}ath(K^*, n)$, we let $p = \{p_0, \ldots, p_n\}$, where $p_0 = \epsilon$ and the nodes are ordered by the successor relation. For any such path p we know that

$$\mathrm{wt}(t, r) \otimes \mathrm{wt}_c(t, \overline{r}) \leq \mathrm{wt}(\overrightarrow{r(t, p_n)}) \otimes \mathrm{wt}_c(\overrightarrow{\overline{r}(t, p_n)}) \ ,$$

and thus

$$\mathrm{wt}(t, r) \otimes \mathrm{wt}_c(t, \overline{r}) \leq \bigotimes_{p \in \mathcal{P}ath(K^*, n)} \mathrm{wt}(\overrightarrow{r(t, p_n)}) \otimes \mathrm{wt}_c(\overrightarrow{\overline{r}(t, p_n)}) \ .$$

Furthermore,

$$\begin{aligned}
\overline{r}(p_n)(r(p_n)) \otimes \mathrm{wt}(\overrightarrow{r(t, p_n)}) \otimes \mathrm{wt}_c(\overrightarrow{\overline{r}(t, p_n)}) \\
&= \overline{\left(\overline{\overline{r}(p_n)(r(p_n))} \oplus \overline{\mathrm{wt}(\overrightarrow{r(t, p_n)})}\right) \otimes} \\
&\qquad \bigotimes_{y=(q_0, t(p_n), q_1, \ldots, q_k)} \left(\overline{\overline{r}(p_n)(q_0)} \oplus \overline{\mathrm{wt}(y)}\right) \oplus \bigoplus_{i \in K} \overline{r}(p_n i)(q_i) \\
&\qquad \text{(by de Morgan's law)} \\
&\leq \overline{\left(\overline{\overline{r}(p_n)(r(p_n))} \oplus \overline{\mathrm{wt}(\overrightarrow{r(t, p_n)})}\right) \otimes} \\
&\qquad \left(\left(\overline{\overline{r}(p_n)(r(p_n))} \oplus \overline{\mathrm{wt}(\overrightarrow{r(t, p_n)})}\right) \oplus \bigoplus_{i \in K} \overline{r}(p_n i)(r(p_n i))\right) \\
&\qquad \text{(choose } y = \overrightarrow{r(t, p_n)}\text{)} \\
&= \overline{r}(p_n)(r(p_n)) \otimes \mathrm{wt}(\overrightarrow{r(t, p_n)}) \otimes \bigoplus_{i \in K} \overline{r}(p_n i)(r(p_n i)) \\
&\qquad \text{(by distributivity of } S\text{)} \\
&= \bigoplus_{i \in K} \overline{r}(p_n)(r(p_n)) \otimes \mathrm{wt}(\overrightarrow{r(t, p_n)}) \otimes \overline{r}(p_n i)(r(p_n i)) \ .
\end{aligned}$$

Using the above two inequations we get

$$\begin{aligned}
&\mathrm{wt}(t, r) \otimes \mathrm{wt}_c(t, \overline{r}) \\
&\quad \leq \left(\bigoplus_{p \in \mathcal{P}ath(K^*, n)} \bigotimes_{j=0}^{n} \overline{r}(p_j)(r(p_j))\right) \otimes \left(\bigotimes_{p \in \mathcal{P}ath(K^*, n)} \mathrm{wt}(\overrightarrow{r(t, p_n)}) \otimes \mathrm{wt}_c(\overrightarrow{\overline{r}(t, p_n)})\right) \\
&\qquad \text{(by induction hypothesis and the first inequation)} \\
&\quad \leq \bigoplus_{p \in \mathcal{P}ath(K^*, n)} \left(\bigotimes_{j=0}^{n-1} \overline{r}(p_j)(r(p_j))\right) \otimes \overline{r}(p_n)(r(p_n)) \\
&\qquad\qquad \otimes \mathrm{wt}(\overrightarrow{r(t, p_n)}) \otimes \mathrm{wt}_c(\overrightarrow{\overline{r}(t, p_n)}) \\
&\qquad \text{(by Lemma 1 b))}
\end{aligned}$$

$$\leq \bigoplus_{p \in \mathcal{P}ath(K^*,n)} \bigoplus_{i \in K} \left(\bigotimes_{j=0}^{n-1} \overline{r}(p_j)(r(p_j)) \right) \otimes \overline{r}(p_n)(r(p_n))$$
$$\otimes \operatorname{wt}(\overrightarrow{r(t,p_n)}) \otimes \overline{r}(p_n i)(r(p_n i))$$

(by the second inequation and distributivity of S)

$$\leq \bigoplus_{p \in \mathcal{P}ath(K^*,n+1)} \bigotimes_{u \in p} \overline{r}(u)(r(u)) \;.$$

(combining p with $p_n i$)

This completes the proof by induction on n. □

This allows us to conclude the first half of the proof of correctness.

Lemma 24. $(\|\overline{\mathcal{A}}\|, t) \leq \overline{(\|\mathcal{A}\|, t)}$.

Proof. Combining Lemmata 22 and 23, we get $\operatorname{wt}(t,r) \otimes \operatorname{wt}_c(t,\overline{r}) \leq 0_S$. Lemma 1 now implies $\operatorname{wt}_c(t,\overline{r}) \leq \overline{\operatorname{wt}(t,r)}$.

Since this holds for all $\overline{r} \in \operatorname{succ}(\overline{\mathcal{A}})$ and all runs r of $\mathcal{A}$, we have $(\|\overline{\mathcal{A}}\|, t) \leq \overline{(\|\mathcal{A}\|, t)}$. □

4.3 Proof of $(\|\overline{\mathcal{A}}\|, t) \geq \overline{(\|\mathcal{A}\|, t)}$

Similar to Theorem 13, we define a successful run $\overline{r} \in Q_c^{K^*}$ of $\overline{\mathcal{A}}$ with $\operatorname{wt}_c(t,\overline{r}) = \overline{(\|\mathcal{A}\|, t)}$.

From Corollary 6 we know that there must be a closed, finite subtree $T \subseteq K^*$ such that for the computation of the weight $(\|\mathcal{A}\|, t)$, we only need to consider the nodes in T.

Definition 25. Let the run $\overline{r} \in Q_c^{K^*}$ be inductively defined as follows:

- $\overline{r}(\epsilon) := \text{in}$.
- If $u \in \operatorname{fr}(T)$ or $u \notin T$, set $\overline{r}(ui) := \underline{0_S}$ for each $i \in K$.
- If $u \in \operatorname{int}(T)$ is a node where $\overline{r}(u)$ has already been defined, set

  $$\overline{r}(ui)(q) := \bigotimes_{\substack{r \in Q^{ui[K^*]} \\ r(ui)=q}} \bigoplus_{w \in ui[K^*] \cap \operatorname{int}(T)} \overline{\operatorname{wt}(\overrightarrow{r(t,w)})}$$

  for each $i \in K$ and $q \in Q$.

From this definition, it is already clear that $\overline{r}$ is a successful run of $\overline{\mathcal{A}}$, since every path will be labeled by $\underline{0_S}$ from some point on.

We additionally define a value $P(u)$ for each node $u \in T$:

$$P(u) := \bigotimes_{r \in Q^{u[K^*]}} \overline{\overline{r}(u)(r(u))} \oplus \bigoplus_{w \in u[K^*] \cap \mathrm{int}(T)} \overline{\mathrm{wt}(\overrightarrow{\overline{r}(t,w)})}$$

Lemma 26. *The following hold:*

- $P(\epsilon) = \overline{(\|\mathcal{A}\|, t)}$.
- $P(ui) = 1_S$ *for all* $ui \in T$.

Proof. The first claim is easily proven by considering the definitions and Corollary 6.

Additionally, for any $ui \in T$ we have

$$\begin{aligned}
P(ui) &= \bigotimes_{r \in Q^{ui[K^*]}} \overline{\overline{r}(ui)(r(ui))} \oplus \bigoplus_{w \in ui[K^*] \cap \mathrm{int}(T)} \overline{\mathrm{wt}(\overrightarrow{\overline{r}(t,w)})} \\
&\geq \bigotimes_{r \in Q^{ui[K^*]}} \overline{\left(\bigoplus_{w \in ui[K^*] \cap \mathrm{int}(T)} \overline{\mathrm{wt}(\overrightarrow{\overline{r}(t,w)})} \right)} \oplus \left(\bigoplus_{w \in ui[K^*] \cap \mathrm{int}(T)} \overline{\mathrm{wt}(\overrightarrow{\overline{r}(t,w)})} \right) \\
&= 1_S \ ,
\end{aligned}$$

which proves the second claim. □

We now show that the run $\overline{r}$ has the claimed weight.

Lemma 27. *The following hold:*

a) $\mathrm{in}_c(\overline{r}(\epsilon)) = 1_S$.

b) $\mathrm{wt}_c(\overrightarrow{\overline{r}(t,u)}) = 1_S$ *for all* $u \notin T$.

c) $\mathrm{wt}_c(\overrightarrow{\overline{r}(t,u)}) = P(u)$ *for all* $u \in T$.

Proof. a) holds by definition of in_c and $\overline{r}(\epsilon)$ and b) follows from the fact that $\overline{r}(u) = \underline{0_S}$ holds for all $u \notin T$. For c), we consider two cases:

- $\mathrm{wt}_c(\overrightarrow{\overline{r}(t,u)}) = P(u)$ for every $u \in \mathrm{fr}(T)$:

$$\begin{aligned}
\mathrm{wt}_c(\overrightarrow{\overline{r}(t,u)}) &= \bigotimes_{y=(q_0,t(u),q_1,\ldots,q_k)} \overline{\overline{r}(u)(q_0)} \oplus \overline{\mathrm{wt}(y)} \\
&= \bigotimes_{r \in Q^{u[K^*]}} \overline{\overline{r}(u)(r(u))} \oplus \overline{\mathrm{wt}(\overrightarrow{r(t,u)})} \\
&= P(u) \ .
\end{aligned}$$

The second equation holds because of idempotency of $\otimes$. We consider any transition y at u as the beginning of every run $r \in Q^{u[K^*]}$ with $\overrightarrow{r(t,u)} = y$.

- $\mathrm{wt}_c(\overrightarrow{\overline{r}(t,u)}) = P(u)$ for every $u \in \mathrm{int}(T)$:

$$\begin{aligned}
&\mathrm{wt}_c(\overrightarrow{\overline{r}(t,u)}) \\
&= \bigotimes_{y=(q_0,t(u),q_1,\ldots,q_k)} \overline{\overline{r}(u)(q_0)} \oplus \overline{\mathrm{wt}(y)} \oplus \bigoplus_{i \in K} \overline{r}(ui)(q_i) \\
&= \bigotimes_{y=(q_0,t(u),q_1,\ldots,q_k)} \overline{\overline{r}(u)(q_0)} \oplus \overline{\mathrm{wt}(y)} \oplus \bigoplus_{i \in K} \bigotimes_{\substack{r_i \in Q^{ui[K^*]} \\ r_i(ui)=q_i}} \bigoplus_{w \in ui[K^*] \cap \mathrm{int}(T)} \overline{\mathrm{wt}(\overrightarrow{r_i(t,w)})} \\
&= \bigotimes_{y=(q_0,t(u),q_1,\ldots,q_k)} \overline{\overline{r}(u)(q_0)} \oplus \overline{\mathrm{wt}(y)} \oplus \\
&\qquad \bigotimes_{\substack{r_1 \in Q^{u1[K^*]} \\ r_1(u1)=q_1}} \cdots \bigotimes_{\substack{r_k \in Q^{uk[K^*]} \\ r_k(uk)=q_k}} \bigoplus_{i \in K} \bigoplus_{w \in ui[K^*] \cap \mathrm{int}(T)} \overline{\mathrm{wt}(\overrightarrow{r_i(t,w)})} \\
&\qquad \text{(by distributivity of } S\text{)} \\
&= \bigotimes_{y=(q_0,t(u),q_1,\ldots,q_k)} \bigotimes_{\substack{r_1 \in Q^{u1[K^*]} \\ r_1(u1)=q_1}} \cdots \bigotimes_{\substack{r_k \in Q^{uk[K^*]} \\ r_k(uk)=q_k}} \\
&\qquad \overline{\overline{r}(u)(q_0)} \oplus \overline{\mathrm{wt}(y)} \oplus \bigoplus_{i \in K} \bigoplus_{w \in ui[K^*] \cap \mathrm{int}(T)} \overline{\mathrm{wt}(\overrightarrow{r_i(t,w)})} \\
&\qquad \text{(by distributivity of } S\text{)} \\
&= \bigotimes_{r \in Q^{u[K^*]}} \overline{\overline{r}(u)(r(u))} \oplus \bigoplus_{w \in u[K^*] \cap \mathrm{int}(T)} \overline{\mathrm{wt}(\overrightarrow{r(t,w)})} \\
&\qquad \text{(concatenate } y \text{ and } r_1, \ldots, r_k \text{ to } r\text{)} \\
&= P(u) \ .
\end{aligned}$$

These two cases account for all the nodes of T, since T is closed. □

This completes the second half of the proof of correctness.

Lemma 28. $(\|\overline{\mathcal{A}}\|, t) \geq \overline{(\|\mathcal{A}\|, t)}$.

Proof. We easily deduce

$$(\|\overline{\mathcal{A}}\|, t) \geq \mathrm{wt}_c(t, \overline{r}) = \mathrm{in}_c(\overline{r}(\epsilon)) \otimes \bigotimes_{u \in K^*} \mathrm{wt}_c(\overrightarrow{\overline{r}(t,u)}) = \overline{(\|\mathcal{A}\|, t)}$$

from Lemmata 26 and 27. □

Theorem 29. *If $\mathcal{A}$ is a WLA and $\overline{\mathcal{A}}$ is its complement automaton, then for all $t \in \Sigma^{K^*}$, $(\|\overline{\mathcal{A}}\|, t) = \overline{(\|\mathcal{A}\|, t)}$.*

Proof. Since the construction of $\overline{\mathcal{A}}$ does not depend on the input tree t, this follows from Lemmata 24 and 28. □

4.4 Complexity

This construction gives us an automaton that has $|S|^{|Q|}$ states, where $|Q|$ is the number of states of the original automaton, and hence can be used to solve the problems $\mathcal{C}_{\mathrm{WL,WB}}$ and $\mathcal{I}_{\mathrm{WL,WB}}$ in exponential time. This is optimal with respect to the complexity of the problems, as shown by Corollary 20.

However, a more fine-grained analysis of the algorithms shows that the black-box approach is in fact more efficient than the glass-box. The main consideration is that the number of meet prime elements of any Boolean lattice is logarithmic in the size of the lattice. Hence, if there are n meet prime elements, then the black-box approach involves n emptiness tests[8] of automata of size $2^{|Q|}$.

On the other hand, the glass-box approach will apply a polynomial time algorithm on an automaton of size $(2^n)^{|Q|}$. Additionally, n cannot be considered independently from $|Q|$, but, given our assumption that the lattice S is generated by the input automata, n actually grows proportionally to $|Q|$. This means that the bigger the input automata become, the more expensive the glass-box approach is, relative to the black-box procedure. This is surprising because it shows that an all-purpose procedure performs better than a specifically designed algorithm.

Obviously, looping automata are not the only ones that can be used in a glass-box approach. In fact, by simply generalizing the construction from Definition 14 to arbitrary Boolean lattices, we could obtain a method for solving $\mathcal{C}_{\mathrm{WC,WB}}$. However, this would again result in an automaton having $|S|^{|Q|}$ states, which is less efficient than the black-box approach.

5 Conclusions

We have investigated some of the standard problems for unweighted automata on infinite trees and their generalization to weighted automata. In particular, we have looked at the inclusion and complementation problems for Büchi automata. Despite this class of automata not being closed under complementation, we have shown that for every looping or co-Büchi automaton it is possible to build a Büchi automaton of exponential size accepting the complement language. We demonstrated that these constructions can be generalized to the weighted setting, thus

[8]The emptiness of Büchi automata can be tested in quadratic time.

giving exponential time solutions to the weighted inclusion and complementation problems. Additionally, we described a black-box approach that solves these problems by performing several (unweighted) inclusion tests.

Since automata on infinite trees provide a clear characterization of reasoning in logics with the tree model property (e.g., some description logics), in our future work we will study the relation between the generalized problems for weighted automata and some non-standard inferences in these logics. In particular, we will study their application to uncertainty and multi-valued reasoning.

Acknowledgements

We are grateful to Christof Löding for sharing his ideas on the complexity of the inclusion problem for Büchi automata.

References

[1] Franz Baader and Rafael Peñaloza. Automata-based axiom pinpointing. *Journal of Automated Reasoning*, 45(2):91–129, August 2010. Special Issue: Selected Papers from IJCAR 2008.

[2] Glenn Bruns and Patrice Godefroid. Model checking with multi-valued logics. In *Proc. of ICALP 2005*, volume 3142 of *Lecture Notes in Computer Science*, pages 281–293. Springer, 2004.

[3] J. Richard Büchi. On a decision method in restricted second order arithmetic. In E. Nagel et al., editors, *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1960.

[4] Nils Buhrke, Helmut Lescow, and Jens Vöge. Strategy construction in infinite games with streett and rabin chain winning conditions. In Tiziana Margaria and Bernhard Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 207–224. Springer Berlin/Heidelberg, 1996.

[5] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. In *Proceedings of ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 2005.

[6] Manfred Droste, Werner Kuich, and George Rahonis. Multi valued MSO logics over words and trees. *Fundamenta Informaticae*, 84(3-4):305–327, 2008.

[7] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata.* Springer, 1st edition, 2009.

[8] Manfred Droste and George Rahonis. Weighted automata and weighted logics on infinite words. In *Proceedings of DLT 2006*, volume 4036 of *Lecture Notes in Computer Science*, pages 49–58. Springer, 2006.

[9] Manfred Droste and Heiko Vogler. Weighted tree automata and weighted logics. *Theor. Comput. Sci.*, 366(3):228–247, 2006.

[10] George Grätzer. *General Lattice Theory.* Birkhäuser, Basel, second edition edition, 1998.

[11] O. Kupferman and Y. Lustig. Lattice automata. In *Proc. 8th International Conference on Verification, Model Checking, and Abstract Interpretation*, volume 4349 of *Lecture Notes in Computer Science*, pages 199 – 213. Springer-Verlag, 2007.

[12] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata and tree automata emptiness. In *STOC '98: Proceedings of the thirtieth annual ACM Symposium on Theory of Computing*, pages 224–233, New York, NY, USA, 1998. ACM.

[13] Salvatore La Torre and Aniello Murano. Reasoning about co-büchi tree automata. In *ICTAC*, pages 527–542, 2004.

[14] Salvatore La Torre, Aniello Murano, and Margherita Napoli. Weak muller acceptance conditions for tree automata. In Agostino Cortesi, editor, *Verification, Model Checking, and Abstract Interpretation*, volume 2294 of *Lecture Notes in Computer Science*, pages 285–288. Springer Berlin/Heidelberg, 2002.

[15] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141(1-2):69–107, 1995.

[16] George Rahonis. Weighted muller tree automata and weighted logics. *Journal of Automata, Languages and Combinatorics*, 12(4):455–483, 2007.

[17] Helmut Seidl. Deciding equivalence of finite tree automata. In B. Monien and R. Cori, editors, *STACS 89*, volume 349 of *Lecture Notes in Computer Science*, pages 480–492. Springer Berlin/Heidelberg, 1989.

[18] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, pages 389–455. Springer New York, 1997.

[19] Moshe Y. Vardi and Thomas Wilke. Automata: From logics to algorithms. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives*, pages 629–736. Amsterdam University Press, 2007.