Technische Universität Dresden
Fakultät für Informatik

Institut für Theoretische Informatik
Prof. Dr.-Ing. Franz Baader

The Infimum Problem as a Generalization of the Inclusion Problem for Automata

Diplomarbeit
zur Erlangung des akademischen Grades

Diplom-Informatiker

vorgelegt von

Name: Borgwardt Vorname: Stefan

geboren am: 06.11.1986 in: Halle/Saale

Tag der Einreichung: 30.09.2010

Betreuer: Dr. rer. nat. Rafael Peñaloza Nyssen

Contents

1 Introduction

Finite-state automata working over any kind of structure are formalisms as old as computer science itself. In particular, we are here concerned with automata over infinite trees. They are given a labeled infinite tree and accept or reject this tree based on its labels. A generalization of these automata with binary decisions are weighted automata. They do not just decide "yes" or "no", but rather compute an arbitrary value from a given algebraic structure, e. g., a semiring or a lattice.

Because of their intimate connection with logical formalisms, automata working over infinite structures are valuable tools in many areas of theoretical computer science, such as model checking (using temporal logics, see [3, 8]) and description logics ([1, 2]). The foundation of most of these applications is the reduction of the satisfiability problem of some logic to the emptiness problem for appropriate automata. Weighted automata can be used to compute priorities of different models or the most "desirable" model according to some specification.

When passing from unweighted to weighted formalisms, many problems can be translated accordingly. There are two basic approaches to lift an algorithm for an unweighted problem to a solution to the corresponding weighted problem. The black-box approach reduces the weighted problem to several applications of the unweighted problems and uses the existing algorithm to solve these. The glass-box approach takes the original algorithm and develops a completely new algorithm based on a generalization of the underlying ideas to the weighted formalism.

The purpose of this work is to determine the feasibility of solving the inclusion problem for automata on infinite trees and its generalization to weighted automata, the infimum aggregation problem. This is basically a sequel to [4], where the same problem has already been considered, albeit not very successful. The most difficult step of both the inclusion problem and the infimum aggregation problem is the complementation of automata. The inclusion problem can be reduced to complementation using polynomial-time constructions and a similar reduction can be done for the infimum aggregation problem. Hence there is a whole chapter in this work dedicated to complementation constructions for different automata models.

This work is structured as follows. After introducing the basic definitions in Chapter 2, we take a look at several problems and constructions for weighted and unweighted automata in Chapter 3. This chapter also contains a complexity analysis of the inclusion problem for Büchi tree automata. Following this, we take a look at the complementation problem for other kinds of tree automata in Chapter 4. This includes constructions for the complementation of unweighted and weighted tree automata with different acceptance conditions and comparisons of glass-box and black-box approaches. We summarize our findings in Chapter 5 and make suggestions for future work in this area.

2 Preliminaries

In this section we first define some general conventions and afterwards introduce the basic notions about trees, lattices and automata.

For an infinite sequence $x := (x_n)_{n\in\mathbb{N}}$ over an alphabet Σ,

$$\mathrm{Inf}(\alpha) := \{\alpha \in \Sigma \mid \exists^\infty n \in \mathbb{N} : x_n = \alpha\}$$

denotes the set of symbols occurring infinitely often in x. For convenience, any sequence $x = (x_n)_{n\in\mathbb{N}}$ may at any time be used as the set $\{x_n\}_{n\in\mathbb{N}}$.

We write $\underline{a}^{X,Y} : X \to Y : x \mapsto a$ for the constant function on some domain X that always yields the value a from some set Y. If X and Y are understood from the context, the superscripts may be dropped from this notation.

In order to shorten many expressions, we define the scope of quantifiers and big operators to always be as large as possible. For instance, in the expression

$$\bigforall_{y\in Y} \bigexists_{x\in X} (f(x) = y \wedge g(x) = 1)$$

one may drop the outer parentheses.

2.1 Trees

Throughout all chapters, $k > 0$ shall be some natural number and $K := \{1, \ldots, k\}$. Our main object of study is the *full k-ary tree* K^*. The elements of K^* are finite sequences over the alphabet K and are called *nodes* or *positions* of the tree K^*. ε denotes the empty sequence and is called the *root node* of K^*. For $i \in K$ and $u \in K^*$, ui is called the *i-th successor of* u (see Figure 2.1). The *full subtree of* K^* *rooted at* u is the set $u[K^*] := \{uv \mid v \in K^*\}$, whose root is u. It is easy to see that $\varepsilon[K^*] = K^*$. The *depth* $|v|$ of a node $v \in u[K^*]$ is the length of the sequence v minus the length of u.

We can define a partial order on K^* as follows: $u \leq w$ iff $w \in u[K^*]$, i. e., iff u is a prefix of w. As usual, $u < w$ abbreviates $u \leq w$ with $u \neq w$. The fact $u \leq w$ is read as "u is above w".

A *subtree* of $u[K^*]$ is a set $T \subseteq u[K^*]$ that is *prefix-closed*, i. e., if $vi \in T$ for $v \in u[K^*]$ and $i \in K$, then $v \in T$ must also hold. The *maximal depth* of T is

$$\mathbf{d}(T) := \begin{cases} -1 & \text{if } T = \emptyset \\ \max\{|v| \mid v \in T\} & \text{if } T \text{ is finite} \\ \infty & \text{otherwise} \end{cases} .$$

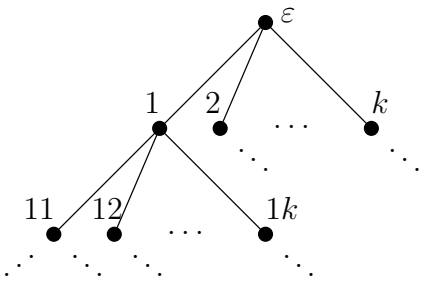

Figure 2.1: The full k-ary tree.

A node $u \in T$ is called *inner node of* T if all its successors ui are also elements of T. A node $u \in T$ is called *leaf of* T if it has no successors inside 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 $\mathrm{fr}(T)$.

A *path* p in $u[K^*]$ is a subtree that contains the root u and for every $v \in p$ has at most one $i \in K$ with $vi \in p$. The *length* of a path is its maximal depth. A path p can be identified with the (finite or infinite) sequence of nodes that starts with u and goes from each node $v \in p$ to its unique successor $vi \in p$, if it exists. For a natural number $0 \leq n \leq \mathbf{d}(p)$, the notation p_n will be used to refer to the $(n+1)$-st node in this sequence, which is the unique node of depth n in p. A path p is *maximal in a subtree* T if $p \subseteq T$ and p is either infinite or its last node has no successor inside T. The set of maximal paths in a subtree T will be denoted by $\mathcal{P}ath(T)$. The set of paths of length $m \in \mathbb{N}$ inside T will be denoted by $\mathcal{P}ath(T, m)$.

For a labeling alphabet Σ, a *($\Sigma-$)labeled tree* is a mapping $K^* \to \Sigma$. As usual, the set of all such mappings is denoted by Σ^{K^*}. Similarly, *(finite) (Σ-)labeled subtrees* are mappings $T \to \Sigma$ where T is a (finite) subtree of $u[K^*]$ for some $u \in K^*$.

2.2 Lattices

A *lattice* $(S, \oplus, \otimes)$ is an algebraic structure on the set S that is equipped with two binary operations *supremum* $\oplus$ and *infimum* $\otimes$. These operations must be commutative and associative and the aborption laws $a \oplus (a \otimes b) = a = a \otimes (a \oplus b)$ must hold for any $a, b \in S$. From these axioms it follows that infimum and supremum are idempotent operations. The infimum of a finite subset $T \subseteq S$ (a finite family $(t_i)_{i \in I} \in S^I$) will be denoted by $\bigotimes T$ ($\bigotimes_{i \in I} t_i$), the supremum by $\bigoplus T$ ($\bigoplus_{i \in I} t_i$).

On each lattice a partial order $\leq$ can be defined as follows: $a \leq b :\Leftrightarrow a \oplus b = b \Leftrightarrow a \otimes b = a$ for $a, b \in S$. In fact, lattices can equivalently be defined as posets in which infimum and supremum of any two elements exist. The operations infimum and supremum are monotone in each component with respect to this partial order.

A lattice[1] S is called *distributive* if $a \oplus (b \otimes c) = (a \oplus b) \otimes (a \oplus c)$ and $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ hold for every $a, b, c \in S$. We will usually deal with finite lattices, in which

[1] We will usually represent the lattice by the underlying set and assume that infimum and supremum are implicitly known.

case distributivity is equivalent to *complete distributivity*. This property asserts that for all families $(a_{i,j})_{i\in I, j\in J(i)}$ of elements of S the equations

$$\bigotimes_{i\in I}\bigoplus_{j\in J(i)} a_{i,j} = \bigoplus_{f\in\mathcal{F}}\bigotimes_{i\in I} a_{i,f(i)} \quad \text{and} \quad \bigoplus_{i\in I}\bigotimes_{j\in J(i)} a_{i,j} = \bigotimes_{f\in\mathcal{F}}\bigoplus_{i\in I} a_{i,f(i)}$$

hold, where $\mathcal{F}$ is defined as the set of all mappings $f : I \to \bigcup_{i\in I} J(i)$ with $f(i) \in J(i)$ for all $i \in I$.

In a finite lattice S we can also write $\bigoplus_{i\in I} t_i$ and $\bigotimes_{i\in I} t_i$ for supremum and infimum of an infinite family $(t_i)_{i\in I} \in S^I$. Furthermore, every finite lattice S is *bounded*, i. e., it has a smallest element $0_S := \bigotimes S$ and a largest element $1_S := \bigoplus_S$.

A bounded lattice S is called *complemented* if for every $a \in S$, there exists a *complement* $b \in S$ with $a \otimes b = 0_S$ and $a \oplus b = 1_S$. If S is distributive, then this complement is unique and we denote the complement of $a \in S$ by $\overline{a}$. A complemented distributive lattice is also called a *Boolean lattice*.

Example 2.1 The lattice $\mathbb{B} := (\{0,1\}, \oplus, \otimes)$ with $0 \leq 1$ is a finite Boolean lattice. It is the smallest nontrivial lattice and is usually used for Boolean logics. In this case, 0 and 1 are interpreted as the truth values *false* and *true*, respectively. The infimum $\otimes$ is the binary conjunction of truth values, $\oplus$ is the disjunction. The expression "*a implies b*" can be represented as $a \leq b$ or equivalently as $\overline{a} \oplus b = 1$ (see Lemma 2.2 a)). ◁

In a Boolean lattice, $\overline{\overline{a}} = a$, $\overline{a \oplus b} = \overline{a} \otimes \overline{b}$ and $\overline{a \otimes b} = \overline{a} \oplus \overline{b}$ hold for all $a, b \in S$. We always have $a \leq b$ iff $\overline{b} \leq \overline{a}$, $\overline{1_S} = 0_S$ and $\overline{0_S} = 1_S$. In a finite Boolean lattice the following equations hold for any family $(t_i)_{i\in I}$ of lattice elements:

$$\overline{\bigotimes_{i\in I} t_i} = \bigoplus_{i\in I} \overline{t_i} \quad \text{and} \quad \overline{\bigoplus_{i\in I} t_i} = \bigotimes_{i\in I} \overline{t_i}.$$

An element p of a lattice S is called *meet prime* if $a \otimes b \leq p$ always implies $a \leq p$ or $b \leq p$. The dual notion is that of a *join prime* element which is defined dually. In a distributive lattice, meet prime elements are exactly the *meet irreducible* elements. These are defined as elements $p \in S$ for which $a \otimes b = p$ always implies $a = p$ or $b = p$. Every element of a distributive lattice S can uniquely be identified by the set of meet prime elements above it, since

$$a = \bigotimes \{p \in S \mid p \text{ meet prime and } a \leq p\}$$

holds for all $a \in S$. In a Boolean lattice S, the complement $\overline{a}$ of a meet prime element $a \in S$ is join prime and the other way around.

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

Lemma 2.2 *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:*

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

Proof:

a) 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 $a \not\leq b$. 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 S is uniquely complemented.

b) By distributivity of S, we have

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

For an alphabet Σ and a lattice S, a *formal tree series over* Σ *and* S is a mapping $\Sigma^{K^*} \to S$. Each labeled tree is thus assigned a value from S. For a formal tree series $f : \Sigma^{K^*} \to S$, one usually writes (f, t) for the image of a tree $t \in \Sigma^{K^*}$ under f and calls this the *coefficient of* f *at* t. The class of all formal tree series over Σ and S is denoted by $S\langle\langle \Sigma^{K^*} \rangle\rangle$.

For convenience, we define complex operations $f \oplus g$, $f \otimes g$ and $\overline{f}$ on tree series $f, g \in S\langle\langle \Sigma^{K^*} \rangle\rangle$ as follows:

$$\begin{aligned} (f \oplus g, t) &:= (f, t) \oplus (g, t) \\ (f \otimes g, t) &:= (f, t) \otimes (g, t) \\ (\overline{f}, t) &:= \overline{(f, t)} \end{aligned}$$

2.3 Automata

We want to look at *(non-deterministic) weighted tree automata* that work on labeled trees $t \in \Sigma^{K^*}$ as input and output a value from some lattice S, i. e., they effectively compute a formal tree series over Σ and S. These automata take the form of a 6-tuple $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt}, \mathfrak{F})$ where Q is a finite set of *states*, Σ is the *input alphabet*, S is a 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 $\mathfrak{F} \subseteq Q^{\mathbb{N}}$ is the *acceptance condition*, described as a predicate on infinite Q-sequences.

A *run* of this automaton is a Q-labeled tree $r \in Q^{K^*}$. Similarly, a *subrun* of $\mathcal{A}$ is a Q-labeled subtree $r \in Q^T$ for a node $u \in K^*$ and a subtree $T \subseteq u[K^*]$. Given a subrun $r \in Q^T$, an input tree $t \in \Sigma^{K^*}$ and an inner node u of T, we define the *transition of* r *on* t *at* u as the tuple $\overrightarrow{r(t,u)} := (r(u), t(u), r(ui), \ldots, r(uk))$. The *weight of* r *(on* t*)* is the value

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

A run $r \in Q^{K^*}$ is called *successful* if for every path $p \in \mathcal{P}ath(K^*)$ the sequence $(r(p_n))_{n \in \mathbb{N}}$ is an element of $\mathfrak{F}$. The set of all successful runs of $\mathcal{A}$ is denoted by $\mathrm{succ}(\mathcal{A})$. The automaton $\mathcal{A}$ defines $\|\mathcal{A}\| \in S\langle\langle \Sigma^{K^*} \rangle\rangle$, the *behavior of* $\mathcal{A}$ (or *tree series recognized by* $\mathcal{A}$) by assigning each input tree $t \in \Sigma^{K^*}$ the value

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

This value is also called the *behavior of* $\mathcal{A}$ *on* t.

Automata are classified by a sequence of several letters. Usually, this sequence has the form $W\alpha A$, where α is determined according to the acceptance condition $\mathfrak{F}$ as follows:

- If $\mathfrak{F} = Q^{\mathbb{N}}$, which is equivalent to no restriction on the runs, then the automaton is called a *looping automaton* and $\alpha := L$. In this case, $\mathcal{A}$ is represented by the 5-tuple $(Q, \Sigma, S, \mathrm{in}, \mathrm{wt})$.
- If there is a set $F \subseteq Q$ of *final states* such that $\mathfrak{F} = \{q \in Q^{\mathbb{N}} \mid \mathrm{Inf}(q) \cap F \neq \emptyset\}$, then the automaton is called a *Büchi automaton* and α is set to B. In the tuple representing the automaton, $\mathfrak{F}$ is then usually replaced by F.

  A *generalized Büchi* condition is expressed by a family $(F_i)_{i \in I}$ of final state sets with a finite index set I. An infinite path $q \in Q^{\mathbb{N}}$ is then accepted if it is accepted by all of the Büchi conditions F_i. α is then replaced by GB.
- The *co-Büchi* condition is similarly defined based on a set $F \subseteq Q$ for which $\mathfrak{F} = \{q \in Q^{\mathbb{N}} \mid \mathrm{Inf}(q) \setminus F = \emptyset\}$. α is then set to C.

  A *generalized co-Büchi* condition is determined by a family $(F_i)_{i \in I}$ of final state sets on a finite index set I. $q \in Q^{\mathbb{N}}$ is accepted if it is accepted by any of the co-Büchi conditions F_i. α is set to GC.
- Another acceptance condition that is based on a set $F \subseteq Q$ is the *weak Büchi* condition, where $\mathfrak{F} = \{q \in Q^{\mathbb{N}} \mid q \cap F \neq \emptyset\}$. This is expressed by $\alpha := b$.
- The *weak co-Büchi* condition is defined by a set $F \subseteq Q$ as $\mathfrak{F} = \{q \in Q^{\mathbb{N}} \mid q \setminus F = \emptyset\}$. α is set to c.
- A *Rabin* condition is based on a set of pairs $\mathcal{F} = \{(E_i, F_i) \mid i \in I\}$ for some finite index set I. A sequence $q \in Q^{\mathbb{N}}$ is then in $\mathfrak{F}$ iff $\mathrm{Inf}(q) \cap F_i \neq \emptyset$ and $\mathrm{Inf}(q) \cap E_i = \emptyset$ for some $i \in I$. This is indicated by $\alpha = R$.
- A *Rabin chain* condition is a Rabin condition $\{(E_1, F_1), \ldots, (E_n, F_n)\}$ $(n \in \mathbb{N})$ for which the strict inclusions $E_1 \subsetneq F_1 \subsetneq E_2 \subsetneq F_2 \subsetneq \ldots \subsetneq E_n \subsetneq F_n$ hold. In this case α is set to RC.
- The *Streett* condition is also defined by a set of pairs $\mathcal{F} = \{(E_i, F_i) \mid i \in I\}$ for a finite set I. $q \in Q^{\mathbb{N}}$ is accepted iff $\mathrm{Inf}(q) \cap F_i = \emptyset$ or $\mathrm{Inf}(q) \cap E_i \neq \emptyset$ for all $i \in I$. $\alpha = S$ indicates this type of acceptance condition.
- A *Muller* condition is given by a set $\mathcal{F} \subseteq \mathcal{P}(Q)$ of state sets. Then $q \in \mathfrak{F}$ iff $\mathrm{Inf}(q) \in \mathcal{F}$. This is indicated by $\alpha = M$.
- A *parity* condition is defined through a function $\pi : Q \to \mathbb{N}$ assigning a *priority* to each state. Based on this, $\mathfrak{F}$ is described as the set $\{q \in Q^{\mathbb{N}} \mid \min \mathrm{Inf}(\pi(q)) \text{ is even}\}$. This acceptance condition is indicated by $\alpha = P$.

The relationships between these acceptance conditions are depicted in Figure 2.2. In this diagram the different conditions are ordered by expressiveness, i. e., the inclusion of the respective classes of recognized tree series. Most of these inclusions are easy consequences of the above definitions. See [12] for the equivalence of Rabin, Streett, Muller and parity conditions.

Example 2.3 We want to illustrate these relationships by a few examples.

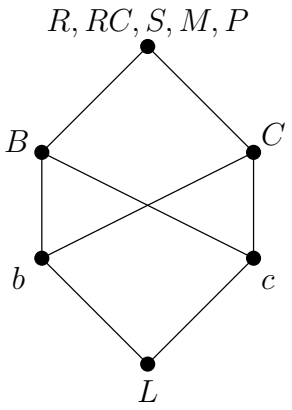

Figure 2.2: The hierarchy of acceptance conditions for weighted tree automata

Let Q be a state set and $F \subseteq Q$ the final state set of a Büchi acceptance condition. Then an equivalent parity condition is defined by the mapping

$$\pi(q) := \begin{cases} 1 & \text{if } q \notin F \\ 0 & \text{if } q \in F \end{cases} .$$

For a parity condition given by $\pi : Q \to \mathbb{N}$, an equivalent Strett condition consists of the pairs

$$(\pi^{-1}(\{0, \ldots, 2k\}), \pi^{-1}(2k+1))$$

for all $k \in \mathbb{N}$ with $0 \leq 2k \leq \max \pi(Q)$. Thus the number of Streett pairs is roughly half the number of priorities. ◁

Example 2.4 Now we want to show that Rabin chain conditions and parity conditions determine equivalent classes of automata (see [12, Definition 6.4]).

Similarly to the previous example, any parity condition $\pi : Q \to \mathbb{N}$ on a state set Q can be expressed as a Rabin chain condition using the pairs

$$(\pi^{-1}(\{0, \ldots, 2k-1\}), \pi^{-1}(\{0, \ldots, 2k\}))$$

for all $k \in \mathbb{N}$ with $0 \leq 2k \leq \max \pi(Q)$.

The converse is also true, since for a Rabin chain condition $E_1 \subsetneq F_1 \subsetneq \ldots \subsetneq E_n \subsetneq F_n$ we find an equivalent parity condition with the priority function

$$\pi(q) := \begin{cases} 2k & \text{if } q \in F_k \setminus E_k \text{ for some } k \\ 2k+1 & \text{if } q \in E_{k+1} \setminus F_k \text{ for some } k \end{cases} ,$$

if we define $F_0 := \emptyset$ and $E_{k+1} := Q$.

Again, the difference between the number of priorities and the number of pairs is only a constant factor of 2. ◁

There are a few other important classes of automata that arise from the following restrictions.

If $S = \mathbb{B}$, the automaton is called *unweighted* and the letter W is dropped out of the classification. In this case, $S = \mathbb{B}$ is usually left out of the tuple describing the automaton. The tree series recognized by $\mathcal{A}$ can then be interpreted as the set

$$\mathcal{L}(\mathcal{A}) := \{t \in \Sigma^{K^*} \mid (\|\mathcal{A}\|, t) = 1\},$$

called the *tree language recognized by* $\mathcal{A}$. If $t \in \mathcal{L}(\mathcal{A})$, then we say that the tree t is *accepted* by $\mathcal{A}$. The functions in and wt are usually replaced by the equivalent sets $I \subseteq Q$ and $\Delta \subseteq Q \times \Sigma \times Q^k$, respectively.

In addition, any classification may be prefixed by the letter D, indicating that the automaton is *deterministic*, i. e., there is exactly one $q \in Q$ for which $\mathrm{in}(q) > 0_S$ and for every $q \in Q$ and $\alpha \in \Sigma$ there is exactly one tuple $(q_1, \ldots, q_k) \in Q^k$ such that $\mathrm{wt}(q, \alpha, q_1, \ldots, q_k) > 0_S$. This implies that for each input tree $t \in \Sigma^{K^*}$ there is exactly one run $r \in Q^{K^*}$ with $\mathrm{wt}(t, r) > 0_S$.

The classes of tree series recognized by a specific type of automata are defined as

$$S^\tau\langle\langle\Sigma^{K^*}\rangle\rangle := \{\|\mathcal{A}\| \mid \mathcal{A} \text{ is a weighted tree automaton of type } \tau\}.$$

In the next chapter we will take a closer look at some of these automata models. We will consider several tasks for weighted tree automata and try to solve them.

3 Weighted Tree Automata

In this chapter, we want to introduce the main problem that we try to solve subsequently. The main focus of this work lies on weighted Büchi automata (WBA), because for this relatively simple acceptance condition there are efficient algorithms that solve a number of fundamental problems.

The next section gives an overview over the basic tasks for Büchi automata. Afterwards we will introduce the main problem and prove a complexity result.

3.1 Basic Results

In this section we will take a look at various closure properties of the class $S^{\mathrm{WBA}}\langle\langle\Sigma^{K^*}\rangle\rangle$. We will later use these to give algorithms for more complex problems by reducing them to a very simple problem - the emptiness problem for Büchi automata. The name of this problem suggests the domain of unweighted automata.

Task (Emptiness Problem)

Given a BA $\mathcal{A}$, decide whether $\mathcal{L}(\mathcal{A}) = \emptyset$. ◁

This can be decided in quadratic time in the size of the input automaton ([10]). The generalization of this problem to weighted automata is as follows.

Task (Behavior Computation)

Given a WBA $\mathcal{A}$ over a singleton alphabet, compute $\|\mathcal{A}\|$. ◁

This can be solved by lifting the unweighted algorithm to the weighted case. In [2] this algorithm was developed and it was shown that its complexity is polynomial in the size of the input automaton and the underlying lattice, if the lattice is finite and distributive. In the remainder of this work S is assumed to be finite and distributive unless specified otherwise.

By reduction to the emptiness problem, several other problems can be shown to be of polynomial complexity.

Task (Infimum Comutation)

Given two WBA $\mathcal{A}_1$, $\mathcal{A}_2$, compute $\|\mathcal{A}_1\| \otimes \|\mathcal{A}_2\|$. ◁

Task (Supremum Computation)

Given two WBA $\mathcal{A}_1$, $\mathcal{A}_2$, compute $\|\mathcal{A}_1\| \oplus \|\mathcal{A}_2\|$. ◁

These problems were solved in [4] by constructing a third WBA $\mathcal{A}$ with $\|\mathcal{A}\| = \|\mathcal{A}_1\| \otimes \|\mathcal{A}_2\|$ (or $\|\mathcal{A}_1\| \oplus \|\mathcal{A}_2\|$), whose behavior could then be computed using the algorithm from [2]. The construction of $\mathcal{A}$ is of polynomial complexity in both cases.

Another result concerning generalized Büchi automata is of interest here. It turns out that $S^{\mathrm{WGBA}}\langle\langle\Sigma^{K^*}\rangle\rangle = S^{\mathrm{WBA}}\langle\langle\Sigma^{K^*}\rangle\rangle$.

Lemma 3.1 *Let $\mathcal{A}$ be a WGBA. Then there is a WBA $\mathcal{A}'$ with $\|\mathcal{A}'\| = \|\mathcal{A}\|$ which is of size polynomial in the size of $\mathcal{A}$.*

Proof: For automata over a singleton alphabet $\Sigma = \{\star\}$, this was proven in [2] by giving an explicit construction for $\mathcal{A}'$. The construction and the proof were lifted to arbitrary weighted automata in [4]. □

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

Lemma 3.2 *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. Then there is a 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.

But since R' is of course still finitely branching, 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 3.3 *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 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 3.2 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 3.4 *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. Then there is a finite subtree $T \subseteq K^*$ such that for every run $r \in Q^{K^*}$ we have*

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

□

Corollary 3.5 *For every unweighted tree automaton $\mathcal{A} = (Q, \Sigma, I, \Delta, \mathfrak{F})$ and every input tree $t \in \Sigma^{K^*}$ there is a finite subtree $T \subseteq K^*$ with the property that*

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

□

3.2 Inclusion

In the remainder of this work, we are concerned with solving the following task.

Task (Infimum Aggregation)

Given two WBA $\mathcal{A}$ and $\mathcal{A}'$, compute $\bigotimes_{t \in \Sigma^{K^*}} (\|\mathcal{A}\|, t) \oplus \overline{(\|\mathcal{A}'\|, t)}$. ◁

This can be seen as a generalization of another task, where both automata are unweighted.

Task (Inclusion Problem)

Given two BA $\mathcal{A}$ and $\mathcal{A}'$, decide whether $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$. ◁

To decide this inclusion is problematic, because one usually checks emptiness of the intersection $\overline{\mathcal{L}(\mathcal{A}')} \cap \mathcal{L}(\mathcal{A})$. However, the class $\mathbb{B}^{\mathrm{BA}}\langle\langle \Sigma^{K^*} \rangle\rangle$ is not closed under complement ([12]), so this approach does not work when one works only with Büchi tree automata. Luckily, the class $\mathbb{B}^{\mathrm{PA}}\langle\langle \Sigma^{K^*} \rangle\rangle$ of all tree languages recognized by unweighted parity automata is closed under complement and intersection ([12]), so we can decide the inclusion problem by viewing $\mathcal{A}$ and $\mathcal{A}'$ as parity automata (see Example 2.3). We will now look at the complexity of this approach.

3.2.1 Complexity of Inclusion

To prepare the ground for the complexity analysis of the inclusion test for unweighted Büchi automata, we need to introduce several new automata models. Since these are only needed in this section, we define them only for the unweighted case.

Definition 3.6 An *automaton on finite trees* is a tuple $\mathcal{A} = (Q, \Sigma, I, \Delta)$, where Q, Σ, I and Δ are defined as for automata on infinite trees. Input trees for these automata are all finite Σ-labeled subtrees. Such a subtree $t \in \Sigma^T$ is *accepted* if there is a finite subrun $r \in Q^T$ with $r(\varepsilon) \in I$ and $\overrightarrow{r(t,u)} \in \Delta$ for every $u \in \mathrm{int}(T)$. The language recognized by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) := \{t \in \Sigma^T \text{ finite labeled subtree} \mid \mathcal{A} \text{ accepts } t\}$. ♢

Definition 3.7 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 non-deterministic 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 pursue 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 are all accepted by $\mathfrak{F}$.

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 3.8 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. The converse of this reduction does not hold. ◁

We are now ready to show the EXPTIME-completeness of the inclusion problem.

Theorem 3.9 *The inclusion problem is EXPTIME-complete.*[1]

Proof: We show EXPTIME-hardness 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 [11, Theorem 2.1] that this problem is EXPTIME-complete.

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

- $\Sigma' := \Sigma \cup \{\star\}$, where $\star$ is a new symbol
- $Q' := Q \cup \{q_\star\}$, where $q_\star$ is a new state
- $\Delta' := \Delta \cup \{(q, \star, q_\star, \ldots, q_\star) \mid q \in Q'\}$
- $F := \{q_\star\}$

[1] Thanks to Christof Löding for the proof idea.

In this construction every finite tree $t \in \Sigma^T$ is uniquely represented by an infinite tree $t' \in \Sigma'^{K^*}$ with $t'(u) = \star$ for every $u \in K^* \setminus T$ and $\mathcal{B}$ accepts only those infinite trees that represent a finite tree in this way. 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 Büchi automata.

We will now give an algorithm that decides $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$ in time exponential in the size of the Büchi automata $\mathcal{A}$ 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 3.8, the equivalent parity condition as in Example 2.3. This construction yields an automaton with n states and 2 priorities.

2) We use [15, 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 (see Example 2.3). 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 a construction from [5, 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 Strett automata and Rabin chain automata (see, e. g., [12]). Rabin chain conditions can equivalently be expressed as parity conditions of the same size (see Example 2.4).

   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'.

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 [7, 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 $\mathcal{A}$ and thus also exponential in the size of both $\mathcal{A}$ and $\mathcal{A}'$. □

3.2.2 From Inclusion to Infimum Aggregation

There are two approaches to get an algorithm for the infimum aggregation problem based on an inlcusion test algorithm. These are general methods that can be used when one is given an unweighted algorithm and wants to transform it into a weighted one.

[2] In [15] alternating automata are defined differently, but the two descriptions can be transformed into each other using only polynomial space.

Glass-box Approach

The so-called *glass-box approach* uses the specifics of the unweighted algorithm and transforms them piece by piece into a weighted version of the algorithm. This is a rather laborious approach which usually only works if the unweighted algorithm is constructive in the first place, i. e., does not prove a complexity result without giving an explicit construction.

Since the inclusion test from the previous section is rather complicated, this cannot easily be applied to our current case. However, we will later use this approach to show that the complement of a WLA-recognizable tree series is recognizable by a WbA (see Section 4.2). For this, we use a rewritten version of the infimum aggregation value:

$$\bigotimes_{t\in\Sigma^{K*}} (\|\mathcal{A}\|, t) \oplus \overline{(\|\mathcal{A}'\|, t)} = \overline{\bigoplus_{t\in\Sigma^{K*}} \overline{(\|\mathcal{A}\|, t)} \otimes (\|\mathcal{A}'\|, t)}$$

One can see that infimum aggregation can also be done directly by complementing the automaton $\mathcal{A}$ and computing the *supremum aggregation* of the resulting two automata. The supremum aggregation problem for WBA was shown to be solvable by a polynomial construction in [4]. We have thus reduced the infimum aggregation to complementation of WBA, similar to the reduction of the inclusion test to the complementation problem for BA.

However, there is still the problem that BA are not closed under complementation, and thus neither are WBA. We will later circumvent this problem by considering other classes of automata for which the complement is a Büchi automaton.

Black-box Approach

The *black-box approach* on the other hand, does not need to know anything about how the unweighted algorithm works. It reduces the weighted problem to one or more problems of the unweighted type and solves these using the "black-box" that is the unweighted algorithm.

In [4] a black-box algorithm for the infimum aggregation was described, which is based on an approach from [6]. For every meet prime element p of the finite Boolean lattice S, one can decide whether $\bigotimes_{t\in\Sigma^{K*}} (\|\mathcal{A}\|, t) \oplus \overline{(\|\mathcal{A}'\|, t)} \leq p$ holds by using the inclusion test on two unweighted automata that are generated from $\mathcal{A}$ and $\mathcal{A}'$. The solution to the infimum aggregation problem is then easily computed as the infimum of all those p for which that test succeeded. We will now describe this procedure in more detail.

For given WBA $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt}, F)$ and $\mathcal{A}' = (Q', \Sigma, S, \mathrm{in}', \mathrm{wt}', F')$ and a meet prime element $p \in S$, the *cropped automata* $\mathcal{A}_p$ and $\mathcal{A}'_{\overline{p}}$ are defined as the BA $(Q, \Sigma, I, \Delta, F)$ and $(Q', \Sigma, I', \Delta', F')$, respectively, where the new initial state sets and transition relations are defined 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 $\mathcal{A}$ ($\mathcal{A}'$). This property is transferred to the behavior of the weighted automata as follows. For any input tree $t \in \Sigma^{K^*}$, we have

$$\begin{aligned}(\|\mathcal{A}\|, t) \leq p &\Leftrightarrow \bigwedge_{r \in \mathrm{succ}(\mathcal{A})} \mathrm{wt}(t, r) \leq p \\ &\Leftrightarrow \bigwedge_{r \in \mathrm{succ}(\mathcal{A})} \mathrm{in}(r(\varepsilon)) \leq p \vee \bigvee_{u \in K^*} \mathrm{wt}(\overrightarrow{r(t, u)}) \leq p \\ &\Leftrightarrow t \notin \mathcal{L}(\mathcal{A}_p)\end{aligned}$$

and similarly, $(\|\mathcal{A}'\|, t) \geq \overline{p} \Leftrightarrow t \in \mathcal{L}(\mathcal{A}'_{\overline{p}})$. Thus,

$$\begin{aligned}\bigotimes_{t \in \Sigma^{K^*}} (\|\mathcal{A}\|, t) \oplus \overline{(\|\mathcal{A}'\|, t)} \leq p &\Leftrightarrow \bigvee_{t \in \Sigma^{K^*}} (\|\mathcal{A}\|, t) \leq p \wedge (\|\mathcal{A}'\|, t) \geq \overline{p} \\ &\Leftrightarrow \bigvee_{t \in \Sigma^{K^*}} t \notin \mathcal{L}(\mathcal{A}_p) \wedge t \in \mathcal{L}(\mathcal{A}'_{\overline{p}}) \\ &\Leftrightarrow \mathcal{L}(\mathcal{A}'_{\overline{p}}) \not\subseteq \mathcal{L}(\mathcal{A}_p).\end{aligned}$$

Since this involves all meet-prime elements of the Boolean lattice S and their number is the same as the size of a generating set of S, this black-box approach will hardly add to the complexity of the unweighted inclusion test. Also note that the construction did not use the acceptance condition of the automata $\mathcal{A}$ and $\mathcal{A}'$, i. e., the same method works for any kind of weighted automata for which the corresponding inclusion problem can be decided.

4 Complementation

We now want to consider a modified infimum aggregation problem.

Task (Infimum Aggregation with WXA)

Given a WXA $\mathcal{A}$ and a WBA $\mathcal{A}'$, compute $\bigotimes_{t \in \Sigma^{K^*}} (\|\mathcal{A}\|, t) \oplus \overline{(\|\mathcal{A}'\|, t)}$. ◁

This task can be approached in two ways, which are both based on a complementation procedure that takes an unweighted XA $\mathcal{A}$ and yields a BA $\overline{\mathcal{A}}$ with $\mathcal{L}(\overline{\mathcal{A}}) = \overline{\mathcal{L}(\mathcal{A})}$. The first approach is to use this for a decision procedure for the inclusion problem with XA and then use the black-box algorithm from the previous section to lift this to a solution of the infimum aggregation problem with WXA.

The second possibility is a glass-box approach that yields a complementation procedure for WXA to WBA. This can then be used to solve the infimum aggregation problem, as detailed in the previous section.

In this chapter we will present several solutions to the following task and compare the complexity of black-box and glass-box approaches based on them.

Task (Complementation for XA)

Given an XA $\mathcal{A}$, construct a BA $\overline{\mathcal{A}}$ with $\mathcal{L}(\overline{\mathcal{A}}) = \overline{\mathcal{L}(\mathcal{A})}$. ◁

For the remainder of this chapter, we fix a finite Boolean lattice S. Since S is isomorphic to some powerset lattice, it is of size $|S| = 2^n$ for some $n \in \mathbb{N}$. There are n meet prime elements in S and thus the black-box approach will need n inclusion tests to solve the infimum aggregation problem.

For the complexity analyses we only compare the size of the state sets of the involved automata. If $\mathcal{A}$ is an unweighted automaton with state set Q, then the size $|\mathcal{A}|$ is $p(|Q|)$ for some polynomial p. Similarly, the size of a weighted automaton with state set Q would be $np(|Q|)$, since we need n bits to store any element of S.[1]

4.1 Complement of deterministic Büchi automata

The first class of automata we want to consider is that of deterministic Büchi automata. In [13, Lemma 8], a construction was presented that yields a BA $\overline{\mathcal{A}}$ recognizing the complement of a language of a given DBA $\mathcal{A}$. The idea behind this is to guess a path in the only valid run of $\mathcal{A}$ that does not fulfill the Büchi condition. This can be done by adding some flags to the states of $\mathcal{A}$. These flags are used to guess a path and a position on this path after

[1] These figures are only valid for the automata models used in this chapter, i. e., automata with simple acceptance conditions.

which no final state is allowed to occur. The transition relation must of course be adjusted accordingly. The resulting automaton is of size $O(|\mathcal{A}|)$.

We will first follow a glass-box approach and give a complementation construction from DWBA to WBA. Aftwerwards we will compare this with the black-box approach.

Definition 4.1 Let $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt}, F)$ be a DWBA over a finite Boolean lattice S. Then the *complement automaton* $\overline{\mathcal{A}}$ is defined as the WBA $(\overline{Q}, \Sigma, S, \overline{\mathrm{in}}, \overline{\mathrm{wt}}, \overline{F})$, where the components are defined as follows:

- $\overline{Q} := Q \times \{0, 1, 2, 3\}$.
- If $\mathrm{in}(q) = 0_S$, we set $\overline{\mathrm{in}}((q, i)) := 0_S$. Otherwise, we define
$$\overline{\mathrm{in}}((q,i)) := \begin{cases} 1_S & \text{if } i = 1 \\ \overline{\mathrm{in}(q)} & \text{if } i = 3 \\ 0_S & \text{otherwise} \end{cases} .$$
- If $\mathrm{wt}(q_0, \alpha, q_1, \ldots, q_k) = 0_S$, we set $\overline{\mathrm{wt}}((q_0, i_0), \alpha, (q_1, i_1), \ldots, (q_k, i_k)) := 0_S$. Otherwise, we define $\overline{\mathrm{wt}}((q_0, i_0), \alpha, (q_1, i_1), \ldots, (q_k, i_k)) :=$
$$\begin{cases} \overline{\mathrm{wt}(q_0, \alpha, q_1, \ldots, q_k)} & \text{if } i_0 = 1 \text{ and } i_1 = \ldots = i_k = 3 \\ 1_S & \text{if } i_0 = 1 \text{ and one } i_j \ (j \geq 1) \text{ is 1 or 2 (others 0)} \\ 1_S & \text{if } i_0 = 2,\ q_0 \notin F \text{ and one } i_j \ (j \geq 1) \text{ is 2 (others 0)} \\ 1_S & \text{if } i_0 = \ldots = i_k \in \{0, 3\} \\ 0_S & \text{otherwise} \end{cases} .$$
- $\overline{F} := Q \times \{0, 2, 3\}$. ♢

For a given input tree $t \in \Sigma^{K^*}$ we know that there is exactly one run $r_t \in Q^{K^*}$ with $\mathrm{wt}(t, r_t) > 0_S$. That means that the behavior of $\overline{\mathcal{A}}$ on t should be exactly $\overline{\mathrm{wt}(t, r_t)}$ if $r_t \in \mathrm{succ}(\mathcal{A})$. If r_t is not successful the behavior should be 1_S.

Theorem 4.2 (Complementation Theorem (DWBA)) *Let $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt}, F)$ be a DWBA over a finite Boolean lattice S. Then $\|\overline{\mathcal{A}}\| = \overline{\|\mathcal{A}\|}$.*

Proof: The idea behind the construction is that the automaton $\overline{\mathcal{A}}$ guesses a path in r_t that violates the acceptance condition of $\mathcal{A}$. If this is possible, the corresponding run of $\overline{\mathcal{A}}$ has weight 1_S, implying $(\|\overline{\mathcal{A}}\|, t) = 1_S$. Otherwise, $\overline{\mathcal{A}}$ generates a run with weight $\overrightarrow{\mathrm{wt}(r_t(t, u))}$ for every node $u \in K^*$ and one run with weight $\overline{\mathrm{in}(r_t(\varepsilon))}$. By taking the supremum of these run weights, we compute exactly $\overline{(\|\mathcal{A}\|, t)} = \overline{\mathrm{wt}(t, r_t)}$.

To show this, we consider all runs $\overline{r} \in \mathrm{succ}(\overline{\mathcal{A}})$ with $\overline{\mathrm{wt}}(t, \overline{r}) > 0_S$. This property implies that the first components of these runs must have a non-zero weight w. r. t. $\mathcal{A}$ and must thus be equal to r_t. We now classify all these runs $\overline{r}$ according to their second components:

a) The second component of every label $\overline{r}(u)$ is 3. There is exactly one such run $\overline{r}$ and its weight is $\overline{\mathrm{in}(r_t(\varepsilon))}$.

b) There is a finite path $p \in \mathcal{P}ath(K^*, n)$ such that $\overline{r}(u)_2 = 1$ for each $u \in p$, $\overline{r}(u)_2 = 3$ for each $u > p_n$ and $\overline{r}(u)_2 = 0$ otherwise. For every finite path $p \in \mathcal{P}ath(K^*, n)$ there is one such run $\overline{r}$ and its weight is $\overrightarrow{\mathrm{wt}(r_t(t, p_n))}$.

c) There is an infinite path $p \in \mathcal{P}ath(K^*)$ and a node $v \in p$ such that $\overline{r}(u)_2 = 1$ for each $u \in p$ with $u \leq v$, $\overline{r}(u)_2 = 2$ for each $u \in p$ with $u > v$ and $\overline{r}(u)_2 = 0$ otherwise. The

first component of all those nodes of p labeled with 2 cannot be a final state, which means that r_t is not successful. Such runs $\overline{r}$ exist iff r_t is not successful and their weight is 1_S.

We conclude

$$\begin{aligned}
(\|\overline{\mathcal{A}}\|, t) &= \bigoplus_{\overline{r}\in\mathrm{succ}(\overline{\mathcal{A}})} \overline{\mathrm{wt}}(t,\overline{r}) \\
&= \bigoplus_{\substack{\overline{r}\in\mathrm{succ}(\overline{\mathcal{A}})\\ \text{of type a)}}} \overline{\mathrm{wt}}(t,\overline{r}) \oplus \bigoplus_{\substack{\overline{r}\in\mathrm{succ}(\overline{\mathcal{A}})\\ \text{of type b)}}} \overline{\mathrm{wt}}(t,\overline{r}) \oplus \bigoplus_{\substack{\overline{r}\in\mathrm{succ}(\overline{\mathcal{A}})\\ \text{of type c)}}} \overline{\mathrm{wt}}(t,\overline{r}) \\
&= \begin{cases} \overline{\mathrm{in}(r_t(\varepsilon))} \oplus \bigoplus_{u\in K^*} \overline{\mathrm{wt}(\overrightarrow{r_t(t,u)})} & \text{if } r_t \in \mathrm{succ}(\mathcal{A}) \\ 1_S & \text{otherwise} \end{cases} \\
&= \begin{cases} \overline{\mathrm{wt}(t, r_t)} & \text{if } r_t \in \mathrm{succ}(\mathcal{A}) \\ 1_S & \text{otherwise} \end{cases} \\
&= \overline{(\|\mathcal{A}\|, t)}.
\end{aligned}$$

□

The size of the state set of the automaton $\overline{\mathcal{A}}$ from Definition 4.1 is still only $O(|Q|)$. The cost of the infimum aggregation for a DWBA $\mathcal{A}$ and a WBA $\mathcal{A}'$ would thus be polynomial in $|Q||Q'|$.

Since the size of the state set of the complement automaton in the unweighted case is also $O(|Q|)$, the black-box approach for the infimum aggregation would involve n inclusion tests which are polynomial in $|Q||Q'|$. One can see that there is no significant difference between the two approaches, since the complexity of the weighted complementation procedure is the same as in the unweighted case.

4.2 Complement of looping automata

In this section we look at a different, but still rather restricted class of automata. First, we will complement any LA into a bA (and thus a BA) and show the correctness of this approach. Afterwards, we will present the corresponding construction for WLA and again compare this with the black-box approach.

4.2.1 Complement of LA

Definition 4.3 For an LA $\mathcal{A} = (Q, \Sigma, I, \Delta)$, define the *complement automaton* $\overline{\mathcal{A}}$ as the bA $(\overline{Q}, \Sigma, \overline{I}, \overline{\Delta}, \overline{F})$ with

- $\overline{Q} := \mathcal{P}(Q)$
- $\overline{I} := \{I\}$
- $(Q_0, \alpha, Q_1, \ldots, Q_k) \in \overline{\Delta} :\Longleftrightarrow$

$$\bigvee_{q_0\in Q_0} \bigvee_{y=(q_0,\alpha,q_1,\ldots,q_k)\in\Delta} \bigvee_{i\in K} q_i \in Q_i$$

- $\overline{F} := \{\emptyset\}$ $\diamondsuit$

The idea is that, for every possible run of $\mathcal{A}$, the automaton $\overline{\mathcal{A}}$ guesses a path violating the transition relation Δ. It aggregates all states that belong to a run for which we have not yet found such a counterexample into sets and will be successful iff all of these sets become empty at some point.

Theorem 4.4 (Complementation Theorem (LA)) *Let $\mathcal{A} = (Q, \Sigma, I, \Delta)$ be an LA. 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 $\overline{r} \in \overline{Q}^{K^*}$ 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 $u = \varepsilon$ we have $r(\varepsilon) \in I = \overline{r}(\varepsilon)$.
- 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 $(\overline{r}(u), t(u), \overline{r}(u1), \ldots, \overline{r}(uk)) \in \overline{\Delta}$. By definition of $\overline{\Delta}$, there must be an $i \in K$ with $r(ui) \in \overline{r}(ui)$. We now append ui to the path p and continue.

Now $\overline{r}$ 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 3.5, there must be a finite subtree $T \subseteq K^*$ on which no valid subrun exists. W.l.o.g. we can assume that every node in T has exactly k successors in T or none at all. We now inductively construct a successful run $\overline{r} \in \overline{Q}^{K^*}$ of $\overline{\mathcal{A}}$ on t for which every node $u \in T$ has the following property:

$$P(u) \quad \equiv \quad \bigwedge_{r \in Q^{u[K^*]}} \Big[r(u) \in \overline{r}(u) \Rightarrow \Big(\bigvee_{w \in u[K^*] \cap T} \overrightarrow{r(t,w)} \notin \Delta \Big) \Big]$$

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 T that lies below u.

- If we set $\overline{r}(\varepsilon) := \{I\}$, then $P(\varepsilon)$ holds because of Corollary 3.5.
- 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.
  2) the properties $P(ui)$ are not violated.

  We do this by expanding $\overline{r}(ui)$ without violating $P(ui)$ while checking the conditions of $\overline{\Delta}$ 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 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 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 $\overline{\Delta}$.

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 $\overline{\Delta}$. If the transition fully lies in T or $\overline{T}$, this is clear from the construction. Let now u be a leaf of T. Since $P(u)$ holds, all mappings $r \in Q^{u[K^*]}$ must violate Δ below u in T and thus at u itself. That means that there cannot be a valid transition of $\mathcal{A}$ at u starting from any $q_0 \in \overline{r}(u)$ and thus the transition $\overrightarrow{\overline{r}(t,u)} = (\overline{r}(u), t(u), \emptyset, \ldots, \emptyset)$ is valid in $\overline{\Delta}$.

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 $t \in \mathcal{L}(\overline{\mathcal{A}})$. □

4.2.2 Complement of WLA

We now augment the construction from the previous section to work with an arbitrary Boolean lattice S. For this the powerset 2^Q is replaced by the set S^Q of all functions mapping the states of $\mathcal{A}$ to lattice values. The other parts of the complement automaton similarly arise from adapting the old definitions to the more general setting.

Definition 4.5 For a WLA $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt})$, define the *complement automaton* as the WbA $\overline{\mathcal{A}} = (\overline{Q}, \Sigma, S, \overline{\mathrm{in}}, \overline{\mathrm{wt}}, \overline{F})$ with

- $\overline{Q} := S^Q$
- $\overline{\mathrm{in}}(\varphi) := \begin{cases} 1_S & \text{if } \varphi(q) \geq \mathrm{in}(q) \text{ for all } q \in Q \\ 0_S & \text{otherwise} \end{cases}$
- $\overline{\mathrm{wt}}(\varphi_0, \alpha, \varphi_1, \ldots, \varphi_k) := \bigotimes_{y=(q_0,\alpha,q_1,\ldots,q_k) \in Q \times \{\alpha\} \times Q^k} \overline{\varphi_0(q_0)} \oplus \overline{\mathrm{wt}(y)} \oplus \left(\bigoplus_{i \in K} \varphi_i(q_i) \right)$
- $\overline{F} := \{\underline{0_S}\}$ where $\underline{0_S} : Q \to S : q \mapsto 0_S$ ♢

We now fix a WLA $\mathcal{A} = (Q, \Sigma, S, \mathrm{in}, \mathrm{wt})$ and an input tree $t \in \Sigma^{K^*}$. To show the correctness of the above construction, we need to show that $(\|\overline{\mathcal{A}}\|, t) = \overline{(\|\mathcal{A}\|, t)}$ holds. The next two sections are dedicated to the two halves of this proof.

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

In order to prove this, we define a successful run $\overline{r} \in \overline{Q}^{K^*}$ of $\overline{\mathcal{A}}$ with the t-weight $\overline{(\|\mathcal{A}\|, t)}$.

From Corollary 3.3 we know that there must be a 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. W. l. o. g. we can assume that every node in T has exactly k successors or none at all, implying that $T = \mathrm{int}(T) \cup \mathrm{fr}(T)$, i. e., every node of T is either a leaf or an inner node of T.

Definition 4.6 Let the run $\overline{r} \in \overline{Q}^{K^*}$ be inductively defined as follows:

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

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

  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 $u \in T$:

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

Lemma 4.7 *The following hold:*

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

Proof: The first claim is easily proven by considering the definitions and Corollary 3.3.

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

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

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

Lemma 4.8 *The following hold:*

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

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

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

Proof: a) holds by definition of $\overline{\mathrm{in}}$ and $\overline{r}(\varepsilon)$ 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:

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

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

  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 $\overrightarrow{r(t,u)} = y$.

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

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

□

This completes the first half of the proof of correctness.

Lemma 4.9 $(\|\overline{\mathcal{A}}\|, t) \geq \overline{(\|\mathcal{A}\|, t)}$.

Proof: Combining Lemmata 4.7 and 4.8, we get

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

□

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

We show this direction by proving the inequality $\overline{\mathrm{wt}}(t, \overline{r}) \leq \overline{\mathrm{wt}(t, r)}$ for all $\overline{r} \in \mathrm{succ}(\overline{\mathcal{A}})$ and $r \in Q^{K^*}$. For $\overline{\mathrm{wt}}(t, \overline{r}) = 0_S$ or $\mathrm{wt}(t, r) = 0_S$ this is trivially satisfied, so we now assume that $\overline{\mathrm{wt}}(t, \overline{r}) > 0_S$ and $\mathrm{wt}(t, r) > 0_S$ hold.

We proceed by showing that $\overline{\mathrm{wt}}(t, \overline{r}) \times \mathrm{wt}(t, r)$ is smaller than $a \otimes \overline{a} = 0_S$ for some suitably chosen $a \in S$. Looking at Theorem 4.4 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 4.10 *There is a depth $m \in \mathbb{N}$ such that*

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

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 be a path of length m in K^* and assume that $\overline{\mathrm{wt}}(t, \overline{r}) \not\leq \bigoplus_{u \in p} \overline{\overline{r}(u)(r(u))}$. 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.□

Lemma 4.11

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

holds for all $n \in \mathbb{N}$.

Proof: For $n = 0$ we have

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

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

Let now the inequation hold for some $n \in \mathbb{N}$. For $p \in \mathcal{P}ath(K^*, n)$, we know that

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

and thus

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

Furthermore,

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

Using the above inequations we get

$$\begin{aligned}
&\mathrm{wt}(t,r) \otimes \overline{\mathrm{wt}}(t,\overline{r}) \\
&\qquad \leq \left(\bigoplus_{p\in \mathcal{P}ath(K^*,n)} \bigotimes_{j=0}^{n} \overline{r}(p_j)(r(p_j))\right) \otimes \left(\bigotimes_{p\in \mathcal{P}ath(K^*,n)} \mathrm{wt}(\overrightarrow{r(t,p_n)}) \otimes \overline{\mathrm{wt}}(\overrightarrow{\overline{r}(t,p_n)})\right) \\
&\qquad\qquad \text{(by induction hypothesis and the first inequation)} \\
&\qquad \leq \bigoplus_{p\in \mathcal{P}ath(K^*,n)} \left(\bigotimes_{j=0}^{n-1} \overline{r}(p_j)(r(p_j)) \otimes \overline{r}(p_n)(r(p_n)) \otimes \mathrm{wt}(\overrightarrow{r(t,p_n)}) \otimes \overline{\mathrm{wt}}(\overrightarrow{\overline{r}(t,p_n)})\right) \\
&\qquad\qquad \text{(by Lemma 2.2 b))} \\
&\qquad \leq \bigoplus_{p\in \mathcal{P}ath(K^*,n)} \bigoplus_{i\in K} \left(\bigotimes_{j=0}^{n-1} \overline{r}(p_j)(r(p_j)) \otimes \overline{r}(p_n)(r(p_n)) \otimes \mathrm{wt}(\overrightarrow{r(t,p_n)}) \otimes \overline{r}(p_n i)(r(p_n i))\right) \\
&\qquad\qquad \text{(by the second inequation)} \\
&\qquad \leq \bigoplus_{p\in \mathcal{P}ath(K^*,n+1)} \bigotimes_{u\in p} \overline{r}(u)(r(u)). \\
&\qquad\qquad \text{(combining } p \text{ with } p_n i) \qquad \square
\end{aligned}$$

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

Lemma 4.12 $(\|\overline{\mathcal{A}}\|, t) \leq \overline{(\|\mathcal{A}\|, t)}$.

Proof: Combining Lemmata 4.10 and 4.11, we get $\mathrm{wt}(t,r) \otimes \overline{\mathrm{wt}}(t,\overline{r}) \leq 0_S$. Lemma 2.2 now implies $\overline{\mathrm{wt}}(t,\overline{r}) \leq \overline{\mathrm{wt}(t,r)}$.

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

Theorem 4.13 (Complementation Theorem (WLA)) *Let $\mathcal{A}$ be a WLA over a finite Boolen lattice. Then $\|\overline{\mathcal{A}}\| = \overline{\|\mathcal{A}\|}$.*

Proof: Since $\overline{\mathcal{A}}$ does not depend on the input tree, this follows from Lemmata 4.9 and 4.12.□

Conclusions

We will now see whether a black-box approach or the presented glass-box algorithm is better suited for solving the infimum aggregation problem with WLA.

- The construction from Definition 4.3 yields a BA $\overline{\mathcal{A}}$ with a state set of size $2^{|Q|}$. Since intersection of Büchi automata and the emptiness test for Büchi automata are of polynomial time complexity, the time complexity for the inclusion test is polynomial in $|Q'|2^{|Q|}$. If we then apply the black-box algorithm from Section 3.2.2, we get an additional factor of n.
- The glass-box algorithm from Definition 4.5 yields a WBA $\overline{\mathcal{A}}$ with a state set of size $|S|^{|Q|} = 2^{n|Q|}$. If we use this algorithm to solve the infimum aggregation problem, we would have a time complexity polynomial in $|Q'|2^{n|Q|}$.

In the case of LA the proposed glass-box algorithm is clearly inferior to the naive black-box approach. This in part due to the fact that the complementation construction for unweighted looping automata is already of exponential time complexity. Another reason is that the weighted construction is simply too wasteful since it uses all functions $Q \to S$ as states. It remains an open problem to find a better construction for the complement of WLA that uses a smaller state set.

4.3 Complement of co-Büchi automata

In this section we present an exponential construction yielding a GBA that recognizes the complement of the tree series recognized by a given CA. This construction originates in the Simulation Theorem from [9]. Among other things, this theorem states that any alternating Büchi automaton can be simulated by a non-deterministic Büchi automaton.

To get from a CA to a GBA recognizing the complement, we first express the CA as an ACA (see Example 3.8), then complement it, which is easy for alternating automata. In the process, the acceptance condition is transformed into a Büchi condition. Using [9, Theorem 1.2], we arrive at a GBA recognizing the complement of the original language. This can be simulated by a BA of polynomial size (see Lemma 3.1).

In the following, we present the whole procedure as a self-contained construction and include a new proof which is similar to the proof of Theorem 4.4.

Definition 4.14 For a CA $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$, the *complement automaton* $\overline{\mathcal{A}}$ is the GBA $(\overline{Q}, \Sigma, \overline{I}, \overline{\Delta}, \overline{F}_1, \ldots, \overline{F}_{|F|+1})$ with

- $\overline{Q} \subseteq \mathcal{P}(Q \setminus F) \times \mathcal{P}(F)^{|F|+1}$ where
  $(Q_0, Q_1, \ldots, Q_{|F|+1}) \in \overline{Q} :\Longleftrightarrow$ the sets $Q_1, \ldots, Q_{|F|+1}$ are pairwise disjoint,
- $\overline{I} := \{(I \setminus F, I \cap F, \emptyset, \ldots, \emptyset)\}$,
- $((Q_0^{(0)}, Q_1^{(0)}, \ldots, Q_{|F|+1}^{(0)}), \alpha, (Q_0^{(1)}, Q_1^{(1)}, \ldots, Q_{|F|+1}^{(1)}), \ldots, (Q_0^{(k)}, Q_1^{(k)}, \ldots, Q_{|F|+1}^{(k)})) \in \overline{\Delta}$

$$:\Longleftrightarrow \bigwedge_{j=0}^{|F|+1} \bigwedge_{q \in Q_j^{(0)}} \bigwedge_{(q,\alpha,q_1,\ldots,q_k)\in\Delta} \Big(\bigvee_{\substack{i\in K\\ q_i\in F}} \bigvee_{l=\max\{j,1\}}^{|F|+1} q_i \in Q_l^{(i)} \Big) \vee \Big(\bigvee_{\substack{i\in K\\ q_i\notin F}} q_i \in Q_0^{(i)} \Big),$$

- $\overline{F}_j := \{(Q_0, Q_1, \ldots, Q_{|F|+1}) \in \overline{Q} \mid Q_j = \emptyset\}$ for $j \in \{1, \ldots, |F|+1\}$. $\diamondsuit$

The states of the complement automaton are tuples of state sets. The first set contains only non-final states, while the remainder are disjoint sets of final states. The idea is that for every run of the original automaton the complement automaton guesses a path which violates the acceptance condition. It accepts iff it is able to find such a path for each run.

If the automaton is in state $(Q_0, \ldots, Q_{|F|+1})$ at node u, it guesses for each state $q \in Q_j$ and possible transition $y = (q, t(u), q_1, \ldots, q_k)$ which direction $i \in K$ to take. The corresponding state q_i is then put in the appropriate set, depending on whether it is a final state or not.

If q and q_i are final states, q_i is added to the j-th component or some component with a greater index. This possibility exists to allow the disjointness condition to be satisfied. If the state q_i is required by several different transitions originating from several components j, it suffices to put q_i in the largest of these components to satisfy all the conditions.

In the end it is checked whether in each component and each path we encounter infinitely many empty sets, which is equivalent to checking whether there are infinitely many non-final states in every guessed path.

We will now show the correctness of the construction in two steps.

Lemma 4.15 *Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a CA. 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 $\overline{r} \in \overline{Q}^{K^*}$ 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 $\overrightarrow{\overline{r}(t,u)} \in \overline{\Delta}$, there must be an $i \in K$ such that $r(ui) \in \overline{r}(ui)_j$ holds for some $j \in \{0, \ldots, |F|+1\}$. 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 $j \in \{1, \ldots, |F|+1\}$. Thus $\overline{r}(u)_j$ can never be empty after the node u_1 along the path p, which contradicts the success of $\overline{r}$. $\square$

For this direction, it is easy to see the similarity to the proof of Theorem 4.4. The other direction is also similar. The property $P(u)$ is replaced by a more complex property $\mathbf{Fail}(u)$ and the proof is generally more complex to account for the different components of each state. Instead of Corollary 3.5, we have to use the more general version in Corollary 3.4 for this proof.

Lemma 4.16 *Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a CA. 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 $\overline{r} \in \overline{Q}^{K^*}$ of $\overline{\mathcal{A}}$ on t. For every node $u \in K^*$ the following property $\mathbf{Fail}(u)$ will be satisfied.

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

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

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

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

$\mathbf{Success}(r, u)$ expresses that a run r is "successful below u", i. e., all infinite paths starting from u must contain only finitely many states from $Q \setminus F$. 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 3.4, and thus $\mathbf{Fail}(u)$ is equivalent to a property $\mathbf{Fail}(u, (T_{j,u}))$ for 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 "$w \in T_{j,u}$".[2]

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 labels of r on the subtree $w[K^*]$ with those of r'_w at every such node $w \in K^*$.[3] This run r' would be a valid and successful run of $\mathcal{A}$ on t, which contradicts the assumption $t \notin \mathcal{L}(\mathcal{A})$.

[2] 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.

[3] 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.

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 $T_{j,u} \subseteq u[K^*]$ ($j \in \{1, \ldots, |F|+1\}$). For every $i \in K$ we construct $\overline{r}(ui)$ from $\overline{r}(u)$ in several steps.

- First we determine an index $\widetilde{j} \in \{1, \ldots, |F|+1\}$ with $\overline{r}(u)_{\widetilde{j}} = \emptyset$. Since we will keep the sets $\overline{r}(u)_j$ ($j \in \{1, \ldots, |F|+1\}$) disjoint, there can be at most $|F|$ nonempty sets and thus such an index $\widetilde{j}$ can always be chosen.
- We initially set $\overline{r}(ui) := (\emptyset, \ldots, \emptyset)$ 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 $\overline{\Delta}$. 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 $j \in \{0, \ldots, |F|+1\}$, $q \in \overline{r}(u)_j$ and $y = (q, t(u), q_1, \ldots, q_k) \in \Delta$, 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. Such an index must always exist, as we will show in the following. 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 $r \in Q^{u[K^*]}$ 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 $\overline{\Delta}$ 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 3.4.
  - 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 $t \in \mathcal{L}(\overline{\mathcal{A}})$. □

The above two lemmata now allow us to conclude that the complementation construction is correct.

Theorem 4.17 (Complementation Theorem (CA)) *Let $\mathcal{A}$ be a CA. Then $\mathcal{L}(\overline{\mathcal{A}}) = \overline{\mathcal{L}(\mathcal{A})}$.* □

This time, we will not follow a glass-box approach to develop a complementation construction for WCA. As was already seen in the previous section, the black-box approach is superior if the unweighted construction is already of exponential complexity. Since the construction from Definition 4.14 is a more general version of Definition 4.3, a similar glass-box approach as in the previous section would again lead to an algorithm that is too expensive. It remains an open problem to find a more efficient complementation construction.

4.4 Another Infimum Aggregation Problem

We now want to look at a modified infimum aggregation problem where $\mathcal{A}$ is a Büchi automaton and $\mathcal{A}'$ is a co-Büchi automaton. Luckily, co-Büchi automata exhibit many of the properties we have used for Büchi automata so far: GCA are polynomially equivalent to CA (see Lemma 4.18 below), there are polynomial constructions for union and intersection ([14, Theorem 4]) and emptiness can be checked in polynomial time ([14, Theorem 9]).[4] Although no explicit generalizations to weighted co-Büchi automata exist, these should be as easy to obtain as the corresponding algorithms for weighted Büchi automata (see [2, 4]).

Lemma 4.18 *Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F_1, \ldots, F_m)$ be a GCA and let the CA $\mathcal{A}'$ be defined as $(Q', \Sigma, I', \Delta', F')$ with*

- $Q' := Q \times \{1, \ldots, m\}$,
- $I' := I \times \{1, \ldots, m\}$,
- $\Delta' := \{((q_0, i_0), \alpha, (q_1, i_1), \ldots, (q_k, i_k)) \mid (q_0, \alpha, q_1, \ldots, q_k) \in \Delta \text{ and } i_0 = \ldots = i_k\}$,
- $F' := \bigcup_{i=1}^{m} F_i \times \{i\}$.

Then we have $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.

Proof: Let $t \in \Sigma^{K^*}$ be an input tree and $r \in Q^{K^*}$ be a valid run of $\mathcal{A}$ that is accepted by some F_i $(i = 1, \ldots, m)$. Then the run $r' \in Q'^{K^*}$ defined by $r'(u) := (r(u), i)$ for all $u \in K^*$ is a successful and valid run of $\mathcal{A}'$, because $\mathrm{Inf}(r', p) \setminus F' = (\mathrm{Inf}(r, p) \times \{i\}) \setminus (F_i \times \{i\}) = \emptyset$ holds for all $p \in \mathcal{P}ath(K^*)$.

Also, all valid runs r' of $\mathcal{A}'$ have this form, i. e., have a constant second component $i \in \{1, \ldots, m\}$ in all labels. The first component of a successful and valid run r' of $\mathcal{A}'$ is thus a successful and valid run r of $\mathcal{A}$, because $(\mathrm{Inf}(r, p) \times \{i\}) \setminus (F_i \times \{i\}) = \mathrm{Inf}(r', p) \setminus F' = \emptyset$ and thus $\mathrm{Inf}(r, p) \setminus F_i = \emptyset$ hold for all $p \in \mathcal{P}ath(K^*)$. □

As always, the problem with the inclusion test (or infimum aggregation) lies in the complementation step. CA are not closed under complement ([14, Theorem 5]) and, even worse, there are tree languages recognized by Büchi automata whose complement cannot be recognized by a co-Büchi automaton ([13, Lemma 3]). This means that a construction similar to that in Definition 4.14 is not possible when the roles of Büchi and co-Büchi automata are reversed.

This means that co-Büchi automata are in some sense "weaker" than Büchi automata and their class of recognized languages is "smaller". Due to this fact, the utility of this modified infimum aggregation problem is limited.

[4] In [14], GCA are called *Landweber tree automata*.

5 Conclusions

In this work we introduced the infimum aggregation problem for weighted tree automata and demonstrated several algorithms to solve it for various acceptance conditions. For deterministic Büchi automata, this problem can be solved in P, for non-deterministic looping, Büchi and co-Büchi automata it was shown to be in EXPTIME.

All of these algorithms were based on solutions for the corresponding unweighted inclusion problems. A black-box approach was used to lift these to the unweighted case, adding only a small factor to the overall complexity. A glass-box approach was found to be preferable only in the case of deterministic Büchi automata.

Although the presented glass-box algorithm for the complement of weighted looping automata was too expensive, it nevertheless provided valuable insights into the structure of the unweighted construction. To develop a construction for weighted automata using a glass-box approach one is forced to review the arguments of the unweighted version in more detail. The proof of correctness of the weighted construction also demonstrated several techniques that can be used to lift an unweighted to a weighted argument.

It remains to see whether there are more efficient constructions for the complement of the classes of weighted tree automata presented here. It seems unlikely, since the black-box approach already adds so little to the complexity of the unweighted algorithm.

In order for the presented construction not to stay purely theoretical, we would also need to find interesting applications for the inclusion problem and the infimum aggregation problem for weighted tree automata. Obvious candidates are the various logics that have been found to be equivalent to certain tree automata. The inclusion problem may be used, e. g., to check for subsumption in description logics where this cannot easily be checked by other methods.

Index

Bibliography

[1] Franz Baader, Jan Hladik and Rafael Peñaloza. Automata can show PSPACE results for description logics. *Information and Computation*, 206(9-10):1045–1056, 2008. Special Issue: 1st International Conference on Language and Automata Theory and Applications (LATA 2007).
doi:10.1016/j.ic.2008.03.006.

[2] Franz Baader and Rafael Peñaloza. Automata-based Axiom Pinpointing. To appear.

[3] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[4] Stefan Borgwardt. Computing Infimum and Supremum of the Behaviour of Weighted Büchi Automata on Labeled Trees, 2010. Großer Beleg.

[5] 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.
doi:10.1007/3-540-61042-1_46.

[6] Manfred Droste, Werner Kuich and George Rahonis. Multi Valued MSO Logics over Words and Trees. *Fundamenta Informaticae*, 84(3-4):305–327, 2008.
URL http://iospress.metapress.com/content/j9652453g663425m/

[7] 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. ACM, New York, NY, USA, 1998.
doi:10.1145/276698.276748.

[8] Stephan Merz. Model Checking: A Tutorial Overview. In Franck Cassez, Claude Jard, Brigitte Rozoy and Mark Ryan (Editors), *Modeling and Verification of Parallel Processes*, volume 2067 of *Lecture Notes in Computer Science*, pages 3–38. Springer Berlin/Heidelberg, 2001.
doi:10.1007/3-540-45510-8_1.

[9] 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.
doi:10.1016/0304-3975(94)00214-4.

[10] Michael O. Rabin. Weakly Definable Relations and Special Automata. In *In Proc. Symp. Mathematical Logic and Foundations of Set Theory*, volume 59 of *Studies in Logic and the Foundations of Mathematics*, pages 1–23. Elsevier, 1970.
doi:10.1016/S0049-237X(08)71929-3.

[11] 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.
doi:10.1007/BFb0029009.

[12] Wolfgang Thomas. Languages, Automata, and Logic. In *Handbook of Formal Languages*, volume 3, pages 389–455. Springer New York, 1997.

[13] Salvatore La Torre and Aniello Murano. Reasoning About Co-Büchi Tree Automata. In *ICTAC*, pages 527–542. 2004.
doi:10.1007/978-3-540-31862-0_37.

[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.
doi:10.1007/3-540-47813-2_17.

[15] 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.
URL http://www.cs.rice.edu/~vardi/papers/wal07.pdf

ERKLÄRUNG

Hiermit erkläre ich, dass ich die am heutigen Tag eingereichte Diplomarbeit zum Thema "The Infimum Problem as a Generalization of the Inclusion Problem for Automata" unter Betreuung von Prof. Dr.-Ing. Franz Baader selbstständig erarbeitet, verfasst und Zitate kenntlich gemacht habe. Andere als die angegebenen Hilfsmittel wurden von mir nicht benutzt.

Datum Unterschrift