Probabilistic Logic, Probabilistic Regular Expressions, and Constraint Temporal Logic

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr. rer. nat.)

im Fachgebiet
Informatik

Vorgelegt

von Dipl.-Inf. Thomas Weidner

geboren am 20. April 1984 in Zeitz, Deutschland.

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Manfred Droste (Universität Leipzig)
2. Dr. habil. Benedikt Bollig (LSV, CNRS & ENS de Cachan)

Die Verleihung des akademischen Grades erfolgt mit Bestehen
der Verteidigung am 21. Juni 2016 mit dem Gesamtprädikat
summa cum laude

Contents

Chapter 1

Introduction

A formal language is an abstract concept in theoretical computer science denoting an arbitrary set of words, trees, or even more complex data structures. A particular interesting class of formal languages is composed of the regular, or recognizable, languages. These are the languages which can be described by finite automata. Since their introduction for finite words by McCulloch and Pitts [MP43], finite automata have been generalised to a wide spectrum of different structures including infinite words by Büchi [B60] and Muller [M63], finite trees by Doner [D65; D70] and Thatcher and Wright [TW65; TW68], and infinite trees by Rabin [R69].

In this thesis, we investigate two extended models of formal languages:

Probabilistic series Instead of a binary decision whether an element is contained in a language, probabilistic series assign a probability value to each element.

Data languages The positions of a word or tree are labelled by a fixed number of arbitrary data values from some domain.

Let us outline our research results and the contents of this thesis:

In the first part, we investigate how classical formalisms for specifying formal languages, like regular expressions or monadic second order logic, can be transferred to the probabilistic setting. We give probabilistic variants of both formalisms over finite and infinite words. We show in each case that our probabilistic regular expressions and probabilistic MSO logic are expressively equivalent to probabilistic automata. In the case of finite trees it turns out that the standard, top-down, probabilistic automaton model is not powerful enough to capture probabilistic MSO logic. Thus, we introduce bottom-up probabilistic tree automata, which are strictly more expressive than the top-down model. See Section 1.1 for details on this.

In the second part, we turn to languages of infinite, multi-dimensional data words, i.e., infinite words with a fixed number of data values at each position. We study linear temporal logic over data words where each data value is a position in the infinite tree. We give a reduction of the model checking problem for this logic to the emptiness problem of constraint Büchi automata. Thereafter, we show that this

problem can be solved in space polynomial in the dimension and logarithmic in the size of the automaton. This implies PSPACE-completeness of the model checking problem for constraint LTL. An extended introduction can be found in Section 1.2.

1.1 Specification of Probabilistic Series

Since the beginning of research on formal languages investigating other formalisms, besides finite automata, to specify regular languages has always been a key topic. In 1956 Kleene [K56] introduced regular expressions and showed that this formalism allows us to specify the same class of languages as finite automata. Instead of giving a machine-like specification, regular expressions permit the specification of a language from finite sets using the operations set union, concatenation, and Kleene-iteration, i.e., the concatenation of a language with itself arbitrarily often. Regular expressions have been generalised to finite trees by Thatcher and Wright [TW68], to the weighted setting by Schützenberger [S61]. More recently, weighted regular expressions have been extended to finite trees by Droste, Pech and Vogler [DPV05], and to valuation monoids, which are an extension of semirings, by Droste and Meinecke [DM11]. A probabilistic variant of regular expressions on finite words has been given by Bollig, Gastin, Monmege and Zeitoun [BGMZ12].

Another formalism well-known is monadic second order (MSO) logic. MSO logic is a restricted form of predicate logic allowing only quantification over single positions and sets of positions, but not over relations or even higher order objects. At the beginning of the 1960s Büchi [B60; B62] showed that the class of languages that can be defined using MSO logic is exactly the class of regular languages. This result has later been extended to finite trees by Thatcher and Wright [TW68], and to infinite trees by Rabin [R69].

In 2005 Droste and Gastin [DG05] gave a weighted extension of MSO logic on words and proved its equivalence to weighted automata. This result was extended to finite trees by Droste and Vogler [DV06] and to valuation monoids by Droste and Meinecke [DM10].

Around the same time as Schützenberger, Rabin [R63] investigated probabilistic automata. In this automaton model, the next state is chosen according to a probability distribution. An extended introduction to this model was given by Paz [P71]. Probabilistic automata have proven very successful and have nowadays a broad range of applications including speech recognition [RST96], prediction of climate parameters [MMST02], or randomized distributed systems [CLSV06]. The model defined by Rabin works for finite words only. At the beginning of the 1970s probabilistic automata were extended to finite trees by Magidor and Moran [MM70] and Ellis [E71]. Probabilistic tree automata have plentiful applications in

the field of natural language processing, including parsing, deep language models, and machine translation. In 2005, probabilistic automata were extended to infinite words by Baier and Grösser [BG05]. This concept gained manifold further research interest [BBG08; CSV11; CDH09; CH10; CT12; TBG09].

Contributions

In this thesis, we develop probabilistic variants of MSO logic and regular expressions that work over finite and infinite words, and over finite ranked trees. We prove that these formalisms are equivalent to an appropriate probabilistic automaton model.

We begin Part I with a Nivat-like theorem for probabilistic series in Chapter 3. This result is used later in Chapter 4 and may also be of independent interest. The classical Nivat theorem [N68] characterises rational transductions by decompositions in a regular language and homomorphisms. Nivat characterisations have attracted recent interest [BD15; DP14]. We give a probabilistic variant of this result characterising the behaviours of probabilistic automata by regular languages and homomorphisms using operations like image, preimage, and the application of a simple probability measure. This characterisation works for finite and infinite words. In the case of finite trees it turns out that standard, top-down, probabilistic tree automata are not powerful enough to capture all functions which can be given using such a Nivat-representation. Therefore, we use the more powerful model of bottom-up probabilistic tree automaton. As probabilistic tree automata form a generalisation of deterministic top-down tree automata, bottom-up probabilistic tree automata provide a generalisation of deterministic bottom-up tree automata. Though this approach seems natural, we found only one other reference to this model [L94]. Furthermore, restricting the Nivat-representation, we also obtain a characterisation of top-down probabilistic tree automata. This shows that the bottom-up model is strictly more expressive than the top-down model.

Next, we introduce probabilistic monadic second order logic in Chapter 4. For this, we extend classical MSO logic by a new second order "expected value" quantifier and close the logic under Boolean operations and expected value. Within the scope of such a quantifier $\mathbb{E}_p X$, formulas $x \in X$ are true with constant probability p. Intuitively, this corresponds to choosing a set X by tossing an unfair coin for each position to decide whether this position is included in the set or not. Using this logic one can define additional new operators like a first order expected value quantifier or a probabilistic universal first order quantifier like in weighted MSO logic. We show that the semantics of probabilistic MSO sentences are exactly the functions which can be described by a Nivat-representation. Thus, we obtain expressive equivalence results for probabilistic MSO logic on finite words and probabilistic automata, for probabilistic MSO logic on infinite words and probabilistic Muller-

automata, and for probabilistic MSO logic on finite ranked trees and bottom-up probabilistic tree automata, respectively.

After having investigated probabilistic MSO logic in Part I, we now turn to regular expressions in Part II. In Chapter 6, we introduce probabilistic regular expressions on infinite words. These expressions extend the expressions introduced in [BGMZ12] in two ways: first, we define a suitable probabilistic ω-operator. Intuitively, given a probabilistic series S, $S^\omega(w)$ is the probability that the word w starts with arbitrarily many words from S. Second, we add a placeholder symbol to the syntax. This placeholder marks the points in expressions, where other expressions can be appended. In contrast to variables in regular tree expressions, this placeholder is purely syntactic and does not occur anywhere in the semantics of an expression. We show that our probabilistic regular expressions are expressively equivalent to probabilistic Muller-automata, a model which is expressively equivalent to probabilistic Rabin-automata. Baier and Grösser [BG05] already showed that probabilistic Rabin-automata are strictly more expressive than probabilistic Büchi-automata. Whereas our construction of an automaton from a given expression is based on the ideas in [BGMZ12], we give a new construction for the converse direction, which unambiguously decomposes the runs of the automaton.

In Chapter 7, we introduce a probabilistic variant of regular tree expressions. We keep the approach from the word case of using a restricted sum operator, but instead of the Kleene-star operator, we use a new iteration operator, which we call infinity-iteration. In Kleene iteration, there is a choice at every step to substitute a variable or not, thus the iteration may stop at any point. This choice is removed in infinity iteration: every occurrence of the iterated variable has to be substituted until the variable does not occur any more. This modified iteration can be modelled much simpler probabilistically than Kleene-iteration with its nondeterministic choices since probabilistic automata do not allow nondeterministic choices, only probabilistic ones. We show that our probabilistic regular tree expressions are equivalent to probabilistic tree automata.

Future Research

Future research might look into extending these results to different structures.

Unranked trees do not restrict the branching structure of a tree like ranked trees do. A characterisation of the recognizable languages of unranked trees by MSO logic has been given by Neven and Schwentick [NS02]. In 2011 this result was extended to the weighted setting by Droste and Vogler [DV11]. For regular tree expressions there already exist forest expressions by Bojańczyk [B07], or one could extend unweighted ranked regular tree expressions to the unranked case. None of these concepts directly fit into the probabilistic setting.

Different, interesting structures are infinite ranked trees. Probabilistic tree automata for infinite trees have been given by Carayol, Haddad and Serre [CHS14]. Extending both, probabilistic regular tree expressions and probabilistic MSO logic to infinite trees poses new challenges. For regular tree expressions, there are uncountably many ways to cut an infinite tree into subtrees. Thus, the introduction of measures to regular tree expressions seems necessary. For probabilistic MSO logic there does not seem to be a proper automaton model. All existing models are top-down based, and thus probably not expressive enough to capture probabilistic MSO logic. Nevertheless, one could still get the equivalence to the tree series defined by the Nivat decomposition from Definition 3.12 similar to the proof of Theorems 5.22 and 5.23.

A different notion of probabilistic regular expressions on trees has been given by Monmege [M13]. These expressions use pebbles and are tree-walking. It has been shown by Bojańczyk, Samuelides, Schwentick and Segoufin [BSSS06] that in the unweighted case pebble tree-walking automata are strictly less expressive than regular tree languages. It remains to be seen if this inclusion also holds in the probabilistic case.

Another direction of research might look into fragments of probabilistic regular expressions or MSO logic. There is ongoing research on subclasses of probabilistic automata with better properties regarding decidability. Notable examples are #-acyclic automata by Gimbert and Oualhadj [GO10], and leaktight automata by Fijalkow, Gimbert and Oualhadj [FGO12]. It would be interesting to see how these subclasses translate to our formalisms. Obtaining equivalence results for these formalisms would allow for easy specification of a well-behaved class of probabilistic automata. Conversely, there may be "natural" fragments of probabilistic regular expressions or probabilistic MSO logic, which admit good decidability properties.

1.2 Model Checking LTL over Data Words

Temporal logics like LTL or CTL* are nowadays standard languages for specifying system properties in verification. These logics are interpreted over node labelled graphs, where the node labels (also called atomic propositions) represent abstract properties of a system (for instance, a computer program). Clearly, such an abstracted system state does not in general contain all the information of the original system state. This may lead to incorrect results in model checking.

In order to overcome this weakness, extensions of temporal logics by atomic (local) constraints over some structure $\mathcal{A}$ have been proposed (cf. [Č94; DG08]). For instance, LTL with local constraints is evaluated over infinite words where the letters are tuples over $\mathcal{A}$ of a fixed size. For instance, for $\mathcal{A} = (\mathbb{Z}, <)$, this logic is

standard LTL where atomic propositions are replaced by atomic constraints of the form $X^i x_j < X^l x_k$. This constraint is satisfied by a path π if the j-th element of the i-th letter of π is less than the k-th element of the l-th letter of π.

While temporal logics with integer constraints are suitable to reason about programs manipulating counters, reasoning about systems manipulating push-down stores requires constraints over words over a fixed alphabet and the prefix relation (which is equivalent to constraints over an infinite k-ary tree with descendant/ancestor relations, where k is the fixed size of the push down alphabet). There are numerous investigations on satisfiability and model checking for temporal logics with constraints over the integers (cf. [Č94; BG06; DG08; G09; BP14; CKL]). On the contrary, temporal logics with constraints over trees have not yet been investigated much, although questions concerning decidability of the satisfiability problem for LTL or CTL* with such constraints have been asked for instance in [DG08; CKL13]. A first (negative) result by Carapelle et al. [CFKL15] shows that a technique developed in [CKL13; CKL] for satisfiability results of branching-time logics (like CTL* or ECTL*) with integer constraints cannot be used to resolve the decidability status of satisfiability of temporal logics with constraints over trees.

Contributions

Our goal is to show that satisfiability of LTL with constraints over the trees is decidable. At first, we analyse the emptiness problem of $\mathcal{T}$-constraint automata (cf. [G09; DD07]) where $\mathcal{T}$ is the infinitely branching infinite tree with prefix relation. These automata are Büchi-automata that process (multi-)data words where the data values are elements of $\mathcal{T}$ and applicability of transitions depends on the order of the data values at the current and the next position. Our technical main result shows that emptiness for these automata is NL-complete for fixed dimension and PSPACE-complete if the dimension is part of the input. Having obtained an algorithm for the emptiness problem, we can easily provide algorithms for the satisfiability and model checking problems for LTL with constraints over $\mathcal{T}$. We exactly mimic the automata based algorithms for standard LTL of Vardi and Wolper [VW94] noting that the constraints in the transitions are precisely what is needed to deal with the atomic constraints in the local constraint version of LTL. It follows directly that satisfiability of LTL with constraints over $\mathcal{T}$ and model checking models defined by constraint automata against LTL with constraints over $\mathcal{T}$ is PSPACE-complete.

Finally, we extend our results to the case of constraints over the infinite k-ary tree for every $k \in \mathbb{N}$ by providing a reduction to LTL with constraints over $\mathcal{T}$. Thus, satisfiability and model checking for LTL with constraints over the infinite k-ary tree is also in PSPACE.

In a parallel work Demri and Deters [DD15] showed above mentioned results on satisfiability using a reduction of constraints over trees to constraints over the integers. Even though the main results of both papers seem to coincide, there are major differences.

1. Demri and Deters' result extends to satisfiability of the corresponding version of CTL*, but Demri and Deters do not consider the model checking problem.

2. Demri and Deters' result holds even if the logic is enriched by length constraints that compare the lengths of the interpretations of variables. Since our approach abstracts away the concrete length of words, we cannot reprove this result. On the other hand, we can enrich the logic with constraints using the lexicographic order on the tree as well. Demri and Deters' approach can not deal with this order. Thus, the logics of both papers are incomparable to each other.

3. Demri and Deters conjecture that the (branching-degree) uniform satisfiability problem is in PSPACE. This problem asks, given a formula and $k \in \mathbb{N} \cup \{\infty\}$, whether there is a model with values in the k-ary infinite tree that satisfies the formula. We confirm Demri and Deters' conjecture.

4. Finally, our proof is self-contained. In contrast, Demri and Deters' proof seems to be more elegant and less technical, but this comes at the cost of relying on the decidability result for satisfiability of LTL with constraints over the integers [BP14], which is again quite technical to prove (In fact, our proof can be easily adapted to reprove this result).

Chapters 8 and 9 are joint work with Alexander Kartzow.

Future Research

Our result opens several further research directions. Firstly, Demri and Deters' result on CTL* with constraints over trees does not yield any reasonable complexity bound because the complexity of their algorithm relies on the results of Bojańczyk and Toruńczyk [BT12] on weak monadic second order logic with the unbounding quantifier. Thus, without any progresses concerning the complexity of this logic, Demri and Deters' approach cannot be used to obtain better bounds. In contrast, the concept of $\mathcal{T}$-constraint automata can be easily lifted to a $\mathcal{T}$-constraint tree-automaton model. Complexity bounds on the emptiness problem for this model would directly imply bounds on the satisfiability for CTL* with constraints over $\mathcal{T}$. Thus, investigating whether our techniques transfer to a result on the emptiness problem of $\mathcal{T}$-constraint tree-automata might be a fruitful approach. Secondly, it

may be possible to lift our results to the global model checking problem similar to the work of Bozelli and Pinchinat [BP14] on LTL with constraints over the integers. Finally, it is a very challenging task to decide whether Demri and Deters' result and our result can be unified to a result on LTL with constraints over the tree with prefix order, lexicographic order and length-comparisons (of maximal common prefixes).

Chapter 2

Preliminaries

We establish basic notations and definitions in this chapter and recall standard results that can be found in the literature.

The set of all natural numbers, *starting with* 1, is denoted by $\mathbb{N}$. The set of all non-negative integers is written as $\mathbb{N}_0$. The set of integers is denoted by $\mathbb{Z}$, the set of rational numbers by $\mathbb{Q}$, and the set of real numbers by $\mathbb{R}$.

Given any set M, we denote its power set by $\mathcal{P}(M)$. Furthermore, for any subset $A \subseteq M$, we write $\mathbb{1}_A$ for the *characteristic function* of A, i.e., the function $\mathbb{1}_A \colon M \to \{0,1\}$ defined by $\mathbb{1}_A(m) = 1$ if and only if $m \in A$ for all $m \in M$.

2.1 Words and Automata on Words

2.1.1 Finite and Infinite Words

Any non-empty set is called an *alphabet*. Unless explicitly noted otherwise, we assume that every alphabet is finite. The elements of an alphabet are called *letters* or *symbols*. Let Σ be an alphabet. A finite, possibly empty, sequence $w = a_1 \cdots a_n$ of elements of Σ is called a *finite word*. The *length* of w is $|w| = n$. The *empty word* is denoted by ε and we set $|\varepsilon| = 0$. We denote the set of all finite words over Σ by Σ^* and the set of all non-empty, finite words over Σ by Σ^+, i.e., $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$.

An infinite sequence of elements of Σ is called an *infinite word* or an *ω-word* over Σ. We write Σ^ω for the set of all infinite words over Σ. For convenience, we define $|w| = \infty$ for every $w \in \Sigma^\omega$. Moreover, we set Σ^∞ as the set of finite or infinite words over Σ, i.e., $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$.

We will use of the *set of positions in a word*. Given a finite word $w \in \Sigma^*$ let $\mathrm{pos}(w) = \{1, \ldots, |w|\}$, and for an infinite word $w \in \Sigma^\omega$ we define $\mathrm{pos}(w) = \mathbb{N}$. For a set $\Gamma \subseteq \Sigma$ and a word $w = (a_i)_{i \in \mathrm{pos}(w)}$, we set $\mathrm{pos}_\Gamma(w) = \{i \in \mathrm{pos}(w) \mid a_i \in \Gamma\}$ and $|w|_\Gamma = |\mathrm{pos}_\Gamma(w)|$. If Γ is singleton, we just use the single letter as index, i.e., $|w|_a$ instead of $|w|_{\{a\}}$.

Any subset $L \subseteq \Sigma^\infty$ is called a *formal language* or just a *language*.

Operations on Languages

Given a finite word u and a finite or infinite word v we write uv for the *concatenation* of u and v. The concatenation of two finite words is again a finite word, whereas the concatenation of a finite and an infinite word is an infinite word.

We use the usual rational operations on formal languages. These are union, concatenation, Kleene-iteration, and ω-iteration. The definition of these operations is given below for languages $L \subseteq \Sigma^*$ and $K \subseteq \Sigma^\infty$:

$$\begin{aligned} L \cdot K &= \{uv \in \Sigma^\infty \mid u \in L,\, v \in K\}, \\ L^* &= \textstyle\bigcup_{i \geq 1} L^n = \{u_1 \cdots u_n \in \Sigma^* \mid n \geq 0,\, u_i \in L \text{ for } i = 1, \ldots, n\}, \\ L^\omega &= \{u_1 u_2 \cdots \in \Sigma^\omega \mid u_i \in L \setminus \{\varepsilon\} \text{ for all } i \geq 1\}, \end{aligned}$$

where $L^0 = \{\varepsilon\}$ and $L^{n+1} = L \cdot L^n$. We also define the ω-operator for a single word $w \in \Sigma^+$ by $w^\omega = www \cdots \in \Sigma^\omega$, i.e, w^ω is the single word in the language $\{w\}^\omega$.

Partial Orders on Words

There are two natural orders on words, that we are interested in:

The *prefix order* $\preceq$ on Σ^∞ is defined by $u \preceq v$ if and only if there is a word $w \in \Sigma^\infty$ such that $uw = v$. This order is a partial order, but it is not linear.

To define the *lexicographic order* $\sqsubseteq$ on Σ^∞ we first fix a linear order $\leq_\Sigma$ on Σ. We set $u \sqsubseteq v$ if either $u \preceq v$ or there are words $x \in \Sigma^*$, $u', v' \in \Sigma^\infty$ and letter $a, b \in \Sigma$ such that $u = xau'$, $v = xbv'$, and $a <_\Sigma b$ holds. It can be shown that $\sqsubseteq$ is a linear order on Σ^∞.

Homomorphisms on Words

Let Σ and Γ be two alphabets. We call any function $h\colon \Sigma^\infty \to \Gamma^\infty$ a *homomorphism* if it satisfies $h(\varepsilon) = \varepsilon$, and $h(uv) = h(u)\,h(v)$ for all $u \in \Sigma^*$ and $v \in \Sigma^\infty$. A function $f\colon \Sigma \to \Gamma^*$ can be extended to a homomorphism $f'\colon \Sigma^\infty \to \Gamma^\infty$ by setting $f'(u) = \big(f(u_i)\big)_{i \in \mathrm{pos}(u)}$ for every word $u = (u_i)_{i \in \mathrm{pos}(u)} \in \Sigma^\infty$. It can be shown that f' is the unique homomorphism which extends f. If a homomorphism $h\colon \Sigma^\infty \to \Gamma^\infty$ satisfies $h(a) \in \Gamma$ for all $a \in \Sigma$, we call h a *relabelling*.

Homomorphisms on Σ^* are defined completely analogously by replacing the symbol ∞ with the symbol $*$.

2.1.2 Finite Automata on Words

After stating fundamental definitions on words, we next define automata as acceptors of formal languages of words. The standard notion of a finite automaton only recognizes languages of finite words. Common extensions to infinite words include Büchi-automata and Muller-automata. As we want to deal with languages

containing both finite and infinite words, we use Muller-automata to accept infinite words and add a set of final states to handle finite words.

Definition 2.1. A *Muller-automaton* over Σ is quintuple $A = (Q, T, I, F, \mathcal{R})$ such that

1. Q is a finite, non-empty set – the set of states,
2. $T \subseteq Q \times \Sigma \times Q$ is any relation – the transition relation,
3. $I \subseteq Q$ is any set of states – the set of initial states,
4. $F \subseteq Q$ is any set of states – the set of final states,
5. $\mathcal{R} \subseteq \mathcal{P}(Q)$ is any system of subsets of states – the Muller acceptance condition.

A *run* of A on a word $w = (a_i)_{i \in \mathrm{pos}(w)}$ is a sequence of states $\rho = (q_i)_{i=0}^{|w|}$ such that $(q_{i-1}, a_i, q_i) \in T$ for all $i = 1, \ldots, |w|$. If $|w| < \infty$, we call a run $\rho = (q_i)_{i=0}^{|w|}$ *successful* if $q_0 \in I$ and $q_{|w|} \in F$. For $|w| = \infty$, the run ρ is successful if $q_0 \in I$ and $\inf(\rho) \in \mathcal{R}$, where $\inf(\rho)$ denotes the set of states that occur infinitely often in ρ. The *language* $\mathrm{L}(A)$ accepted by the automaton A consists of all finite and infinite words w such that there exists a successful run of A on w.

An automaton A is called *deterministic* if $|\{q \in Q \mid (p, a, q) \in T\}| \leq 1$ for all $p \in Q$ and $a \in \Sigma$, and $|I| \leq 1$, i.e., in any state, there is at most one possible subsequent state when reading any letter and there is at most one initial state. The automaton A is *complete* if $|\{q \in Q \mid (p, a, q) \in T\}| \geq 1$ for all $p \in Q$ and $a \in \Sigma$, and $|I| \geq 1$, i.e., in any state, there is at least one possible subsequent state when reading any letter and there is at least one initial state. If the automaton A is deterministic and complete, we can replace the transition relation T by the *transition function* $\delta \colon Q \times \Sigma \to Q$, which is uniquely defined by $(p, a, \delta(p, a)) \in T$ for all $p \in Q$ and $a \in \Sigma$.

A language $L \subseteq \Sigma^\infty$ is called *recognizable* or *regular* if there is a Muller-automaton A with $\mathrm{L}(A) = L$.

A *finite automaton* is a Muller-automaton $A = (Q, T, I, F, \mathcal{R})$ with $\mathcal{R} = \emptyset$. We just write $A = (Q, T, I, F)$ in this case. A Muller-automaton on infinite words is a Muller-automaton $A = (Q, T, I, F, \mathcal{R})$ with $F = \emptyset$. We write $A = (Q, T, I, \mathcal{R})$ for a Muller-automaton on infinite words.

Example 2.2. Let $\Sigma = \{\mathrm{a}, \mathrm{b}\}$ and consider the automaton A shown in Fig. 2.1. Note that the depicted automaton is deterministic and complete, i.e., in every state there is exactly one reachable state for every possible edge label.

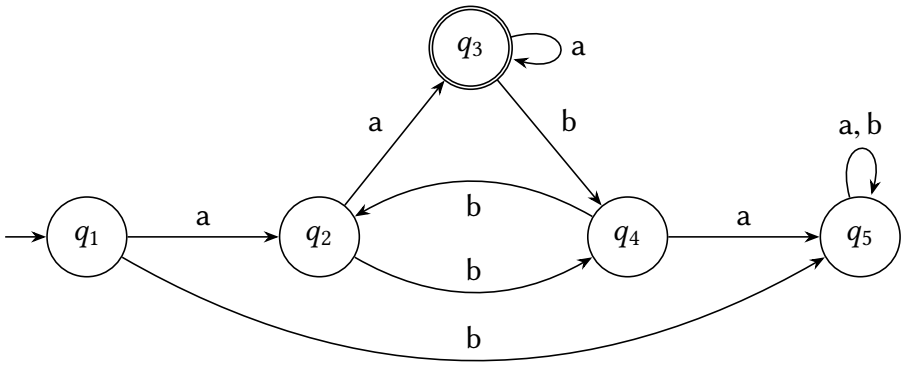


Figure 2.1: The automaton A with $\mathcal{R} = \{\{q_2, q_4\}, \{q_2, q_3, q_4\}\}$

The automaton accepts all finite words of length at least 2 that start and end with the letter a and for which every consecutive sequence of b's has even length. This can be seen as follows: starting from state q_1, the letter b leads to state q_5, which is not accepting and cannot be left. Thus, the first letter of an accepted word must be a. Afterwards, the automaton can read an arbitrary number of a's reaching state q_2 or q_3. In both cases after reading a letter b, another letter b has to follow directly, otherwise the automaton would enter state q_5 from q_4. Thus, the number of consecutive b's has to be even. In order to reach the accepting state q_3 the last letter has to be a. Conversely, the run of every finite word of the claimed form is accepting. For infinite words, the automaton has to stay in the middle triangle forever, visiting at least q_2 and q_4 infinitely often. Thus, every infinite word accepted by A has to contain infinitely many b's. In total we obtain

$$\begin{aligned} \mathrm{L}(A) =& \{ \mathrm{a}u\mathrm{a} \in \Sigma^* \mid u \in \Sigma^*, \text{ every maximal sequence of b's in } u \text{ has even length} \} \\ & \cup \{ \mathrm{a}w \in \Sigma^\omega \mid |w|_\mathrm{b} \text{ is infinite} \}. \end{aligned}$$

The existence of a deterministic Muller-automata recognizing the same language for any non-deterministic ω-automaton on infinite words is a classical result going back to McNaughton [M66]. It was later improved by Safra [S88]. The result still holds if the automaton accepts finite and infinite words, as this case can be reduced to the infinite word case.

Lemma 2.3. Let A be a Muller-automaton. There is a deterministic and complete Muller-automaton A' with $\mathrm{L}(A) = \mathrm{L}(A')$.

Proof. Let $A = (Q, T, I, F, \mathcal{R})$. Furthermore, let $\# \notin \Sigma$ be a new symbol. We define a Muller-automaton on infinite words $A_1 = (Q_1, T_1, I, \mathcal{R}_1)$ over $\Sigma \cup \{\#\}$ where $Q_1 = Q \cup \{q_\#\}$, $\mathcal{R}_1 = \mathcal{R} \cup \{\{q_\#\}\}$, and

$$T_1 = T \cup \{(q_\#, \#, q_\#)\} \cup \{(q, \#, q_\#) \mid q \in F\}.$$

Thus, for every infinite words $w \in \Sigma^\omega$ we have $w \in \mathrm{L}(A)$ if and only if $w \in \mathrm{L}(A_1)$, and for finite words $w \in \Sigma^*$ we conclude $w \in \mathrm{L}(A)$ if and only if $w\#^\omega \in \mathrm{L}(A_1)$. By McNaughton's theorem [M66], there is a deterministic and complete Muller-automaton on infinite words $A_2 = (Q_2, T_2, I_2, \mathcal{R}_2)$ with $\mathrm{L}(A_2) = \mathrm{L}(A_1)$. We obtain the automaton A' by defining $A' = (Q', T', I', F', \mathcal{R}')$ with $Q' = Q_2$, $I' = I_2$, $\mathcal{R}' = \mathcal{R}_2$, $T' = T_2 \cap Q' \times \Sigma \times Q'$, and by letting F' consist of all states $q \in Q'$ such that there is a run ρ of A_2 on $\#^\omega$ starting in q with $\inf(\rho) \in \mathcal{R}_2$. Intuitively, we obtain A' from A_2 by removing all transitions that are labelled with # and by making every state final for which there exists an accepting run on $\#^\omega$ starting in q. One shows that $\mathrm{L}(A') \cap \Sigma^\omega = \mathrm{L}(A_2) \cap \Sigma^\omega$ and $w \in \mathrm{L}(A')$ if and only if $w\#^\omega \in \mathrm{L}(A_2)$ for every finite word $w \in \Sigma^*$. ∎

2.2 Probabilistic Automata and Measures on Words

Our goal is to introduce probabilistic ω-automata as defined by Baier and Grösser [BG05]. For this, we recall some basic probability theory in Section 2.2.1. Readers familiar with probability theory can skip this section. To motivate probabilistic ω-automata we give the definition of probabilistic automata on finite words, as introduced by Rabin [R63], below.

For the rest of this section, let Σ be an alphabet. In the following, let for any finite or countable, non-empty set M, $\Delta(M)$ be the set of all *distributions* on M, i.e., all functions $d \colon M \to [0,1]$ such that $\sum_{m \in M} d(m) = 1$.

Definition 2.4. A *probabilistic automaton* is a quadruple $A = (Q, \delta, \mu, F)$ where

1. Q is a finite, non-empty set – the set of states,
2. $\delta \colon Q \times \Sigma \to \Delta(Q)$ is a function – the transition probability function,
3. $\mu \in \Delta(Q)$ is a distribution – the initial distribution,
4. $F \subseteq Q$ is any subset of states – the set of final states.

We sometimes write $\delta(p, a, q)$ instead of $\delta(p, a)(q)$. The *behaviour of A* is the function $\|A\| \colon \Sigma^* \to [0, 1]$ which is given by

$$\|A\|(w) = \sum_{\substack{q_0,\dots,q_{n-1}\in Q \\ q_n\in F}} \mu(q_0) \prod_{i=1}^{n} \delta(q_{i-1}, w_i, q_i) \tag{2.1}$$

for all $w = w_1 \cdots w_n \in \Sigma^*$. A function $S \colon \Sigma^* \to [0, 1]$ is called *recognizable* if there is a probabilistic automaton A with $\|A\| = S$.

Next, we would like give the extension of Definition 2.4 to infinite words. The behaviour of probabilistic ω-automata is a generalisation of (2.1). As there may be uncountably many runs on an infinite word, this behaviour cannot be modelled by the means of a simple sum any more. Instead, one has to make use of measure theory to obtain meaningful semantics. We will only give a brief introduction to measure theory in the next section.

2.2.1 Measures on Words and Runs

For the convenience of the reader, we recall the notions of a σ-algebra and a measure in this section. We state some standard results that we will use later in this work. At the end, we give the Ionescu-Tulcea theorem which is crucial for the definition of probabilistic ω-automata. For a comprehensive introduction into probability theory see, e.g., [K08]. A chapter about topology on finite and infinite words can be found in [PP04].

Definition 2.5. Let Ω be an arbitrary, non-empty set. We make the following definitions:

1. A *σ-algebra* over Ω is system of sets $\mathcal{A} \subseteq \mathcal{P}(\Omega)$, which contains the empty set, and is closed under complement and countable union. An element of $\mathcal{A}$ is called a *measurable set*. The pair $(\Omega, \mathcal{A})$ is called a *measurable space*.

2. Given any system of sets $\mathcal{X} \subseteq \mathcal{P}(\Omega)$, we denote the smallest σ-algebra $\mathcal{A}$ with $\mathcal{X} \subseteq \mathcal{A}$ by $\sigma(\mathcal{X})$. Given a σ-algebra $\mathcal{A}$, we say that the set $\mathcal{E} \subseteq \mathcal{A}$ *generates* $\mathcal{A}$ if $\sigma(\mathcal{E}) = \mathcal{A}$.

3. A *measure* on a measurable space $(\Omega, \mathcal{A})$ is a function $\mu \colon \mathcal{A} \to \overline{\mathbb{R}_+}$, where $\overline{\mathbb{R}_+} = \mathbb{R}_+ \cup \{\infty\}$, such that $\mu(\emptyset) = 0$ and $\mu(\bigcup_{i\geq 1} A_i) = \sum_{i\geq 1} \mu(A_i)$ for all pairwise disjoint families of measurable sets $A_1, A_2, \dots \in \mathcal{A}$. The triple $(\Omega, \mathcal{A}, \mu)$ is called a *measure space*. In case $\mu(\Omega) < \infty$, we say that μ is *finite*.

   The measure μ is called a *probability measure* if $\mu(\Omega) = 1$. In this case we call $(\Omega, \mathcal{A}, \mu)$ a *probability space*.

4. If Ω is at most countable, we call any function $d\colon \Omega \to [0,1]$ with $\sum_{x\in\Omega} d(x) = 1$ a *probability distribution* or just a *distribution* on Ω. Any distribution uniquely determines a measure μ_d on $(\Omega, \mathcal{P}(\Omega))$ by letting $\mu_d(M) = \sum_{x\in M} d(x)$ for all $M \subseteq \Omega$. To ease notation, we write d for μ_d. Thus, we view every distribution on Ω also as a measure on $(\Omega, \mathcal{P}(\Omega))$.

The set of all distributions on Ω is denoted by $\Delta(\Omega)$.

Our goal is to construct a probability measure on the set of all infinite runs. Therefore, we need to define two things: first, a suitable σ-algebra on the set of runs, which contains the set of accepting runs. Second, a way to construct a unique probability measure from the transition probabilities of an automaton. We also want our definition to be a real extension of the finite word case and thus be able to deal with finite and infinite words.

A standard way to define a σ-algebra is to start with a metric space and consider the σ-algebra generated by the open sets. Hence, we define the usual metric on words. See [PP04] for an extended introduction to topology on the set of words.

Definition 2.6. Let Σ be a finite alphabet. We define the metric d_Σ on Σ^∞ by

$$d_\Sigma(u,v) = \begin{cases} 2^{-\min(D(u,v))} & \text{if } u \neq v \\ 0 & \text{if } u = v, \end{cases}$$

where

$$D(u,v) = \{\, i \in \operatorname{pos}(u) \cap \operatorname{pos}(v) \mid u_i \neq v_i \,\} \cup (\operatorname{pos}(u) \triangle \operatorname{pos}(v)).$$

We will call this metric space just Σ^∞ and assume d_Σ is understood. Moreover, we consider Σ^ω and Σ^* as metric subspaces of $(\Sigma^\infty, d_\Sigma)$.

One easily checks that $(\Sigma^\infty, d_\Sigma)$ is really a metric space. Intuitively, in this metric space words are near to each other if they agree on a long prefix. One can show that Σ^∞ with this metric is a compact metric space, i.e., if it is covered by a family of open sets, then there is already a finite subfamily which also covers the whole space.

Next, we define the notion of the Borel-σ-algebra, which arises from the open sets. Recall that a subset A of a metric space (X,d) is *open* if for every $a \in A$ there is an $\varepsilon > 0$ such that every $x \in X$ with $d(a,x) < \varepsilon$ is also contained in A.

A function $f\colon X \to X'$ between two metric spaces (X,d) and (X',d') is called *continuous* if $f^{-1}(A)$ is open in (X,d) for every open set $A \subseteq X'$.

Definition 2.7. Let (X,d) be a metric space. The *Borel-σ-algebra* $\mathcal{B}(X,d)$ is the smallest σ-algebra such that the open sets of (X,d) are measurable. In other words

$$\mathcal{B}(X,d) = \sigma\big(\{A \subseteq X \mid A \text{ open in } (X,d)\}\big).$$

If d is understood, we write $\mathcal{B}(X)$ for $\mathcal{B}(X, d)$.

Though this definition is the standard definition of the Borel-σ-algebra, one might like a more explicit representation. In the case of words, it suffices to consider so called *cylinder sets*, i.e., sets of the form $u\Sigma^\infty$ for $u \in \Sigma^*$, as generators of the σ-algebra to obtain $\mathcal{B}(\Sigma^\infty)$.

Lemma 2.8. Let Σ be a finite alphabet. The following equations hold:

$$\mathcal{B}(\Sigma^\infty) = \sigma(\{u\Sigma^\infty \mid u \in \Sigma^*\}) \qquad \text{and} \qquad \mathcal{B}(\Sigma^\omega) = \sigma(\{u\Sigma^\omega \mid u \in \Sigma^*\}).$$

Note that the Borel-σ-algebra on Σ^* is just $\mathcal{P}(\Sigma^*)$, as Σ^* is countable.

Proof. We only show the Σ^∞ part, the second equation is analogous. Every set of the form $u\Sigma^\infty$ is open: let $x \in u\Sigma^\infty$ and $y \in \Sigma^\infty$ with $d_\Sigma(x, y) < 2^{-|u|}$. By definition of d_Σ, x and y agree at least on the first $|u|$ letters. Hence, $y \in u\Sigma^\infty$. We conclude $\sigma(\{u\Sigma^\infty \mid u \in \Sigma^*\}) \subseteq \mathcal{B}(\Sigma^\infty)$.

Conversely, let $A \subseteq \Sigma^\infty$ open. By definition of open set and d_Σ there is a $u_x \in \Sigma^*$ for every $x \in A$ such that u_x is a prefix of x and $u_x\Sigma^\infty \subseteq A$. Thus, $A = \bigcup_{x\in A} u_x\Sigma^\infty$. As the set of finite words is countable, there is a countable subset $A' \subseteq A$ which satisfies $A = \bigcup_{x\in A'} u_x\Sigma^\infty$. Therefore, $A \in \sigma(\{u\Sigma^\infty \mid u \in \Sigma^*\})$. Hence, $\mathcal{B}(\Sigma^\infty) \subseteq \sigma(\{u\Sigma^\infty \mid u \in \Sigma^*\})$. ∎

The system of cylinder sets is easier to handle in most cases than the system of all open sets. Nevertheless, the system of cylinder sets is still closed under intersection. Therefore, the values of a measure on these sets uniquely determine the measure, which is a standard result in measure theory.

Lemma 2.9. Let $(\Omega, \mathcal{A})$ be a measurable space and $\mathcal{E} \subseteq \mathcal{A}$ such that $\sigma(\mathcal{E}) = \mathcal{A}$, and $A \cap B \in \mathcal{E}$ for all $A, B \in \mathcal{E}$. Moreover, let μ and ν be finite measures on $(\Omega, \mathcal{A})$ such that $\mu(E) = \nu(E)$ for all $E \in \mathcal{E}$. Then $\mu(A) = \nu(A)$ for all $A \in \mathcal{A}$.

This completes the definition of a suitable σ-algebra on the set of runs. Next, we state the definition of a measurable function. These functions allow us to transfer measures from one σ-algebra to another. Moreover, they are the function we can use to define a meaningful measure integral.

Definition 2.10. Let $(\Omega, \mathcal{A})$ and $(\Omega', \mathcal{A}')$ be measurable spaces. A function $f : \Omega \to \Omega'$ is called $(\Omega, \mathcal{A})$-$(\Omega', \mathcal{A}')$-*measurable*, or just *measurable*, if $f^{-1}(A') \in \mathcal{A}$ for all $A' \in \mathcal{A}'$.

Note that, as the preimage is compatible with union and complement, it suffices to show the relation $f^{-1}(A') \in \mathcal{A}$ for sets $A' \in \mathcal{E}$ where $\mathcal{E} \subseteq \mathcal{A}$ generates $\mathcal{A}$. In particular, given two metric spaces (X, d) and (X', d'), every continuous function $f : X \to X'$ is $\mathcal{B}(X)$-$\mathcal{B}(X')$-measurable.

If f is any function $\Omega \to \Omega'$, then the system $f^{-1}(\mathcal{A}') = \{ f^{-1}(A') \mid A' \in \mathcal{A}' \}$ forms a σ-algebra on Ω. We call $f^{-1}(\mathcal{A}')$ the *σ-algebra generated by* f.

Measurable functions can not only be used to transfer σ-algebras to a different domain, but also measures. We will use the following construction in several proofs.

Proposition 2.11. Let $(\Omega, \mathcal{A})$ and $(\Omega', \mathcal{A}')$ be measurable spaces, $f : \Omega \to \Omega'$ a measurable function, and μ a measure on $(\Omega, \mathcal{A})$. Then, μ' defined by $\mu'(A') = \mu(f^{-1}(A'))$ is a measure on $(\Omega', \mathcal{A}')$. We write $\mu' = \mu \circ f^{-1}$.

Next, we state the Ionescu-Tulcea theorem, which allows us to construct measures on an infinite sequences from so-called transition kernels. Whereas the statement of the theorem works on arbitrary σ-algebras, we only give a variant for finite σ-algebras, i.e., in the following X is always a finite set and we consider $\mathcal{P}(X)$ as the σ-algebra on X.

Theorem 2.12 (Ionescu-Tulcea theorem for finite sets). Let X be a finite set, d a distribution on X, and $\kappa_i : X^2 \to [0, 1]$ be mappings such that $\kappa_i(x, \cdot) \in \Delta(X)$ for each $x \in X$ and every $i \geq 1$. There is a unique probability measure μ on $(X^\omega, \mathcal{B}(X))$ such that

$$\mu(x_0 \cdots x_n X^\omega) = d(x_0) \prod_{i=1}^{n} \kappa_i(x_{i-1}, x_i), \tag{2.2}$$

for all $n \geq 1$ and $x_0, \ldots, x_n \in X$.

Note that Theorem 2.12 only yields a measure on infinite words over X. As μ is already a probability measure, every extension of μ to the finite and infinite words would assign probability 0 to every finite word. This is due to the fact that the kernels κ_i transfer the whole probability mass to the next position of the sequence. The following variant of Theorem 2.12 which deals also with finite words will become useful:

Corollary 2.13. Let Q be a finite alphabet, d a distribution on Q and $\kappa_i : Q^2 \to [0, 1]$ be mappings such that $\sum_{b \in Q} \kappa_i(a, b) \leq 1$ for every $a \in Q$ and $i \geq 1$. There is a unique probability measure μ on $\mathcal{B}(Q^\infty)$ such that

$$\mu(a_0 \cdots a_n Q^\infty) = d(a_0) \prod_{i=1}^{n} \kappa_i(a_{i-1}, a_i), \tag{2.3}$$

for all $a_0 \cdots a_n \in Q^+$.

Proof. Let $\perp \notin Q$ be a new symbol and $X = Q \cup \{\perp\}$. Define transition kernels κ_i' on X by $\kappa_i'(a,b) = \kappa_i(a,b)$, $\kappa_i'(a,\perp) = 1 - \sum_{b \in Q} \kappa_i(a,b)$, $\kappa_i'(\perp,b) = 0$, and $\kappa_i'(\perp,\perp) = 1$ for all $a, b \in Q$ and $i \geq 1$. Moreover, let $d' \in \Delta(X)$ with $d'(a) = d(a)$ and $d'(\perp) = 0$ for all $a \in Q$. With these definitions, d' and $(\kappa_i')_{i \geq 1}$ satisfy the conditions of Theorem 2.12. Thus, there is a probability measure μ' on $\mathcal{B}(X^\infty)$ such that (2.2) holds. Let $\pi : X^\omega \to Q^\infty$ be the homomorphism given by $\pi(q) = q$ for all $q \in Q$ and $\pi(\perp) = \varepsilon$. This function is measurable as $\pi^{-1}(q_1 \cdots q_n Q^\infty) = \{\perp\}^* q_1 \{\perp\}^* \cdots \{\perp\}^* q_n X^\omega$. Explicit calculation shows that the measure $\mu = \mu' \circ \pi^{-1}$ satisfies (2.3). As the system of cylinder sets is closed under intersection, μ is unique. ∎

We recall the definition of the measure integral. Let $\mathcal{B}(\mathbb{R})$ denote the Borel-σ-algebra on the reals, where the usual topology on $\mathbb{R}$ is assumed. We only establish the integral for cases that we actually use later, i.e., positive functions. For a definition on general function see [K08].

Definition 2.14. Let $(\Omega, \mathcal{A}, \mu)$ be a measurable space. We make the following definitions:

1. A function $s : \Omega \to [0, \infty)$ is called *simple* if $s = \sum_{i=1}^n r_i \mathbb{1}_{A_i}$ for some values $r_1, \ldots, r_n \in [0, \infty)$ and measurable sets $A_1, \ldots, A_n \in \mathcal{A}$. We define the integral of s with respect to μ by
$$\int s(x)\, \mu(\mathrm{d}x) = \sum_{i=1}^n r_i \mu(A_i).$$

2. Let $f : \Omega \to [0, \infty)$ be a measurable function. The *integral* of f with respect to μ is given by
$$\int f(x)\, \mu(\mathrm{d}x) = \sup\left\{ \int s(x)\, \mu(\mathrm{d}x) \;\middle|\; s \text{ simple function with } 0 \leq s \leq f \right\}.$$
A shorter notation for the above integral is $\int f \, \mathrm{d}\mu$.
We call f *integrable* if $\int f \, \mathrm{d}\mu < \infty$.

3. For any measurable set $A \in \mathcal{A}$ and measurable function f, we define
$$\int_A f \, \mathrm{d}\mu = \int \mathbb{1}_A \cdot f \, \mathrm{d}\mu.$$

4. If μ is a probability measure, the *expected value* of a measurable function $f : \Omega \to [0, \infty)$ is given by
$$\mathbb{E}[f] = \int f \, \mathrm{d}\mu.$$

To conclude this section, we define finite product spaces and state Fubini's theorem.

Definition 2.15. Let $(\Omega_1, \mathcal{A}_1, \mu_1), \ldots, (\Omega_n, \mathcal{A}_n, \mu_n)$ be measurable spaces. We define the *product σ-algebra* $\mathcal{A}$ of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ by

$$\mathcal{A} = \bigotimes_{i=1}^{n} \mathcal{A}_i = \sigma(\{A_1 \times \cdots \times A_n \mid A_i \in \mathcal{A}_i \text{ for } i = 1, \ldots, n\}).$$

The *product measure* μ of $\mu_1, \ldots, \mu_n$ on $\mathcal{A}$ is then given by

$$\mu(A_1 \times \cdots \times A_n) = \prod_{i=1}^{n} \mu_i(A_i),$$

for all sets $A_i \in \mathcal{A}_i$ for $i = 1, \ldots, n$.

Note that product spaces and preimages can be interchanged with each other: Let $(\Omega_i, \mathcal{A}_i, \mu_i)$ be measure spaces, $(\Omega_i', \mathcal{A}_i')$ be measurable spaces and $f_i \colon \Omega_i' \to \Omega_i$ be measurable functions for $i = 1, \ldots, n$. We define a function $f \colon \Omega_1' \times \cdots \times \Omega_n' \to \Omega_1 \times \cdots \times \Omega_n$ by $f(x_1, \ldots, x_n) = (f_1(x_1), \ldots, f_n(x_n))$. This function is measurable in the corresponding product spaces. Moreover, the following equalities hold:

$$\bigotimes_{i=1}^{n} f_i^{-1}(\mathcal{A}_i) = f^{-1}\left(\bigotimes_{i=1}^{n} \mathcal{A}_i\right) \qquad \text{and} \qquad \bigotimes_{i=1}^{n} (\mu \circ f_i^{-1}) = \left(\bigotimes_{i=1}^{n} \mu_i\right) \circ f^{-1}.$$

Fubini's theorem states that integration in product spaces can be decomposed in two integrations in the corresponding original measure spaces. Furthermore, the order of this decomposition does not matter.

Theorem 2.16 (Fubini's theorem). Let $(\Omega_1, \mathcal{A}_1, \mu_1)$ and $(\Omega_2, \mathcal{A}_2, \mu_2)$ be measure spaces and $f \colon \Omega_1 \times \Omega_2 \to [0, \infty)$ be a measurable function from $\mathcal{A}_1 \otimes \mathcal{A}_2$ to $\mathcal{B}(\mathbb{R})$. Then the functions $x \mapsto \int f(x, y)\, \mu_2(\mathrm{d}y)$ and $y \mapsto \int f(x, y)\, \mu_1(\mathrm{d}x)$ are measurable and the following equalities hold:

$$\begin{aligned} \int f(x, y)\ (\mu_1 \otimes \mu_2)(\mathrm{d}(x, y)) &= \int \left(\int f(x, y)\ \mu_2(\mathrm{d}y) \right) \mu_1(\mathrm{d}x) \\ &= \int \left(\int f(x, y)\ \mu_1(\mathrm{d}x) \right) \mu_2(\mathrm{d}y). \end{aligned}$$

2.2.2 Probabilistic ω-Automata

In this section, we define probabilistic ω-automata. This model has first been defined by Baier and Grösser [BG05]. Our definition extends Baier and Grösser's definition slightly: first, we use a Muller-acceptance condition and not a Rabin-acceptance condition, and second, we allow also the acceptance of finite words by adding a final state set. Moreover, we add sink states to the automaton model, i.e., states where no further transition is possible. The definitions and proofs in Chapter 6 will rely on these modifications. Note that the different choice of the acceptance condition does not change the expressive power of the automata model, cf. Definition 3.6 and Theorem 3.9.

In the following the set $\Delta_0(X)$ contains in addition to all distributions on X, the null function, i.e., $\Delta_0(X) = \Delta(X) \cup \{\mathbb{0}\}$ where $\mathbb{0}(x) = 0$ for all $x \in X$.

Definition 2.17. A *probabilistic Muller-automaton* over an alphabet Σ is a quintuple $A = (Q, \delta, \mu, F, \mathcal{R})$ where

1. Q is a finite, non-empty set – the set of states,
2. $\delta\colon Q \times \Sigma \to \Delta_0(Q)$ – the transition probability function,
3. $\mu \in \Delta(Q)$ – the initial distribution,
4. $F \subseteq Q$ – the set of final states,
5. $\mathcal{R} \subseteq \mathcal{P}(Q)$ – the Muller-acceptance condition.

A state $q \in Q$ is called a *sink* if $\delta(q, a) = \mathbb{0}$ for all $a \in \Sigma$.

For every word $w = (w_i)_{i \in \mathrm{pos}(w)} \in \Sigma^\infty$ let $\Pr_A^w$ be the unique probability measure on $(Q^\infty, \mathcal{B}(Q^\infty))$ given by

$$\Pr_A^w(q_0 \cdots q_n Q^\infty) = \begin{cases} \mu(q_0) \prod_{i=1}^n \delta(q_{i-1}, w_i, q_i) & \text{if } n \leq |w| \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Given a measurable set M, we write $\Pr_A(M)$ for the function $w \mapsto \Pr_A^w(M)$. The set of *successful runs* on words of length $n \in \mathbb{N} \cup \{\infty\}$ is given by

$$\mathfrak{S}_n = \begin{cases} Q^n F & \text{if } n < \infty \\ \{\rho \in Q^\omega \mid \inf(\rho) \in \mathcal{R}\} & \text{if } n = \infty. \end{cases}$$

The *behaviour of* A is the function $\|A\|\colon \Sigma^\infty \to [0,1]$ defined by

$$\|A\|(w) = \Pr_A^w(\mathfrak{S}_{|w|})$$

for every $w \in \Sigma^\infty$.

A function $f\colon \Sigma^\infty \to [0,1]$ is called *recognizable* if there is a probabilistic Muller-automaton A with $\|A\| = f$.

The existence of the measure $\Pr_A^w$ in Definition 2.17 is a direct consequence of Corollary 2.13. Given a word $w = (w_i)_{i\in\mathrm{pos}(w)} \in \Sigma^\infty$, let $\kappa_i(p,q) = \delta(p,w_i,q)$ if $i \le |w|$ and $\kappa_i(p,q) = 0$ if $i > |w|$. This definition of the kernels κ_i satisfy the requirements of Corollary 2.13. Thus, there is a measure μ such that (2.3) holds. By definition of the κ_i this is just (2.4).

Finally, we argue that $\mathfrak{S}_n$ is a measurable set in $\mathcal{B}(Q^\infty)$. For $n \in \mathbb{N}$ we have $\mathfrak{S}_n = Q^nFQ^\infty \setminus Q^{n+2}Q^\infty$. Thus, $\mathfrak{S}_n \in \mathcal{B}(Q^\infty)$. In case $n = \infty$ we can rewrite $\mathfrak{S}_\infty$ using cylinder sets:

$$\mathfrak{S}_\infty = \bigcup_{\{r_1,\dots,r_k\}=R\in\mathcal{R}} \left(\bigcap_{i\ge1} \bigcup_{j\ge i} Q^j r_1 Q^* r_2 \cdots r_{k-1} Q^* r_k Q^\omega \cap \bigcup_{i\ge1} \bigcap_{j\ge i} Q^j R Q^\omega \right).$$

As the set of finite words over Q is countable, this shows $\mathfrak{S}_n \in \mathcal{B}(Q^\infty)$.

Example 2.18. Let $\Sigma = \{\mathrm{a},\mathrm{b}\}$ and $0 < p < 1$. We consider the probabilistic Muller-automaton $A = (Q,\delta,\mu,\emptyset,\mathcal{R})$ from Fig. 2.2a where $\mathcal{R} = \{\{\mathrm{I},\mathrm{F}\},\{\mathrm{F}\}\}$. We show $\|A\|(w) = 1 - p^{|w|_\mathrm{a}}$ by direct computation: let $w = w_1w_2\cdots \in \Sigma^\omega$.

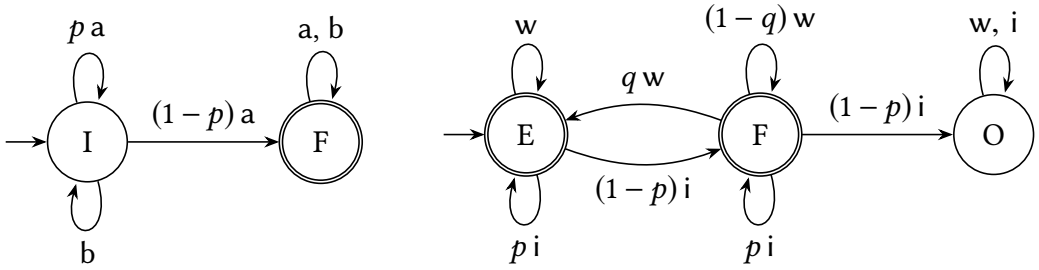
$$\|A\|(w) = \Pr_A^w(\{\rho \in Q^\omega \mid \mathrm{F} \in \inf(\rho)\})$$

By the structure of the automaton, F can not be left with positive probability. Thus $\Pr_A^w(\{\rho \in Q^\omega \mid \mathrm{F} \in \inf(\rho)\}) = \Pr_A^w(\mathrm{I}^*\mathrm{F}^\omega) = \Pr_A^w(\mathrm{I}^*\mathrm{F}Q^\omega)$.

$$\begin{aligned}
&= \sum_{n\ge1} \Pr_A^w(\{q_0q_1\cdots \in Q^\omega \mid q_n = \mathrm{F},\ q_i = \mathrm{I} \text{ for all } i < n\}) \\
&= \sum_{\substack{n\ge1\\ w_i=\mathrm{a}}} (1-p) \prod_{i=1}^{n-1} \begin{cases} p & \text{if } w_i = \mathrm{a} \\ 1 & \text{if } w_n = \mathrm{b} \end{cases} \\
&= \sum_{\substack{n\ge1\\ w_n=\mathrm{a}}} (1-p)\, p^{|w_1\cdots w_{i-1}|_\mathrm{a}} \\
&= 1 - p^{|w|_\mathrm{a}},
\end{aligned}$$

where we set $p^\infty = 0$.

Example 2.19. We consider a communication device for sending messages. At every point of time either a new input message becomes available or the device



(a) Example 2.18 (b) Example 2.19

Figure 2.2: Probabilistic Automata A from Examples 2.18 and 2.19. Transitions without a probability value have probability 1.

is waiting for a new message. When a new message is available, the device tries to send this message. Sending a message succeeds with probability $p \in (0,1)$. In this case the message is stored in an internal buffer. The next time the device is waiting for a new message, sending the stored message is retried. Intuitively, as sending a buffered message has already failed once, it seems to be harder to send this message. So sending a buffered message is only successful with probability $q \in (0,1)$. The buffer can hold one message.

We model this device, using the two letter alphabet $\Sigma = \{\mathrm{w}, \mathrm{i}\}$ for the events "wait" and "input message". The automaton $A = (Q, T, I, \emptyset, \mathcal{R})$ given in Fig. 2.2b assigns to every word $w \in \Sigma^\infty$ the probability that this sequence of "wait" and "input message" events does not overflow the buffer. The automaton has a Büchi acceptance condition, i.e., $\mathcal{R} = \{X \subseteq Q \mid X \cap \{\mathrm{E}, \mathrm{F}\} \neq \emptyset\}$. We chose the states E, F, and O corresponding to the conditions empty buffer, full buffer, and overflow. The transitions model the behaviour explained in the first paragraph.

We consider the language L of words $w \in \Sigma^\omega$ with $\|A\|(w) > 0$, i.e., all event sequences with a positive probability not to overflow the buffer. In contrast to finite words, where the language of words with positive acceptance probability is always regular, the language L is not regular. This can be seen as follows: let $w = uv^\omega$ be an ultimately periodic word with $|v|_\mathrm{i} > 0$. Using Markov chain theory one shows that the probability to be in state O after reading uv^{n+1} is at least $1 - \lambda^n$ for some $\lambda > 0$. Thus, intuitively, the probability to be in state O is 1, after reading uv^ω. Therefore, $\|A\|(w) = 0$. Nevertheless, there are words w with infinitely many letters i and $\|A\|(w) > 0$. Consider a word $w = \mathrm{i}\mathrm{w}^{n_1}\mathrm{i}\mathrm{w}^{n_2}\mathrm{i}\mathrm{w}^{n_3}\cdots$. Using induction one shows $\Pr_A^w(\mathrm{E}\{\mathrm{E},\mathrm{F}\}^{n_1}\mathrm{E}\cdots\mathrm{E}\{\mathrm{E},\mathrm{F}\}^{n_k}Q^\omega) = \prod_{i=1}^{k}(1-(1-p)(1-q)^{n_i})$ for every $k \geq 1$. Therefore, we obtain

$$\|A\|(\mathrm{i}\mathrm{w}^{n_1}\mathrm{i}\mathrm{w}^{n_2}\mathrm{i}\mathrm{w}^{n_3}\cdots) \geq \prod_{i\geq 1}\big(1-(1-p)(1-q)^{n_i}\big).$$

By choosing $n_i = i$, i.e. $\mathsf{iwiw}^2\mathsf{iw}^3 \cdots = u$, we obtain $\|A\|(u) > 0$ since $\sum_{i\geq 1}(1-p)(1-q)^i < \infty$, for details see [K51, chapter 7].

We can conclude that L is not regular: by the last paragraph, we know that L contains at least one infinite word. If L was a regular language, it would also contain a lasso shaped word, but every lasso shaped word has probability zero. Hence, L can not be regular.

We conclude this section with a useful proposition how to decompose $\Pr_A^w$ into a measure on a suffix of w and values of δ.

Proposition 2.20. Let $A = (Q, \delta, \mu, F, \mathcal{R})$ be a probabilistic Muller-automaton and $A_q = (Q, \delta, \mathbb{1}_{\{q\}}, F, \mathcal{R})$ for all $q \in Q$. Furthermore, let $M \subseteq Q^\infty$ measurable, $w \in \Sigma^\infty$, and $n \leq |w|$. The following statement holds:

$$\Pr_A^w(M) = \sum_{q_0,\ldots,q_n \in Q} \mu(q_0) \cdot \left(\prod_{i=1}^{n} \delta(q_{i-1}, w_i, q_i) \right) \cdot \Pr_{A_{q_n}}^{w_{n+1}w_{n+2}\cdots}\left(\left\{\rho \in Q^\infty \mid q_0 \cdots q_n\rho \in M\right\}\right). \quad (2.5)$$

Proof. One checks that for fixed n and fixed states $q_0, \ldots, q_n$ the mapping $M \mapsto \Pr_A^{w_{n+1}w_{n+2}\cdots}\left(\left\{\rho \in Q^\infty \mid q_0 \cdots q_n\rho \in M\right\}\right)$ is a finite measure. Thus, so is the complete right side of (2.5). Let $r_0 \cdots r_m Q^\infty$ be a cylinder set. We have

$$\left\{\rho \in Q^\infty \mid q_0 \cdots q_n\rho \in r_0 \cdots r_m Q^\infty\right\} = \begin{cases} r_{n+1} \cdots r_m Q^\infty & \text{if } q_i = r_i \text{ for all } 0 \leq i \leq n \\ \emptyset & \text{otherwise.} \end{cases}$$

Therefore, we obtain as right side of (2.5):

$$\begin{aligned} \text{r.s. of (2.5)} &= \mu(r_0) \cdot \left(\prod_{i=1}^{n} \delta(r_{i-1}, w_i, r_i) \right) \cdot \Pr_{A_{r_n}}^{w_{n+1}\cdots}(r_{n+1} \cdots r_m Q^\infty) \\ &= \mu(r_0) \cdot \left(\prod_{i=1}^{n} \delta(r_{i-1}, w_i, r_i) \right) \cdot \left(\prod_{j=n+1}^{m} \delta(r_{j-1}, w_j, r_j) \right) \\ &= \Pr_A^w(r_0 \cdots r_m Q^\infty). \end{aligned}$$

As the system of cylinder sets is an intersection closed generating system, the proof is complete. ∎

2.3 Trees and Tree Automata

This section gives the basic definitions regarding trees and tree automata. Note that we only consider finite ranked trees here.

2.3.1 Finite Ranked Trees

A *ranked alphabet* is an alphabet Σ with a function $arity\colon \Sigma \to \mathbb{N}_0$. For every $n \in \mathbb{N}_0$ let $\Sigma_n = \{f \in \Sigma \mid arity(f) = n\}$. We just write Σ instead of $(\Sigma, arity)$ if the function $arity$ is understood.

A *tree* over Σ is a mapping $t\colon D \to \Sigma$, where $D \subseteq \mathbb{N}^*$ such that

1. D is finite and non-empty,
2. D is prefix-closed, i.e., $uv \in D$ implies $u \in D$ for all $u, v \in \mathbb{N}^*$,
3. $\{i \mid xi \in D\} = \{1, \ldots, arity(f)\}$ for all $x \in D$ with $f = t(x)$.

We write $\mathrm{pos}(t)$ for the set D. We identify a symbol $a \in \Sigma_0$ with the tree a' where $\mathrm{pos}(a') = \{\varepsilon\}$ and $a'(\varepsilon) = a$. As in the word case, we set $\mathrm{pos}_A(t) = \{x \in \mathrm{pos}(t) \mid t(x) \in A\}$ for some set $A \subseteq \Sigma$. For singleton sets $A = \{f\}$, we write $\mathrm{pos}_A(t) = \mathrm{pos}_f(t)$. The set of all *leaf positions* $\mathrm{leaf}(t)$ contains all $\preceq$-maximal elements of $\mathrm{pos}(t)$, where $\preceq$ denotes the prefix order. The set of *inner positions* is $\mathrm{inner}(t) = \mathrm{pos}(t) \setminus \mathrm{leaf}(t)$. We denote the set of all trees over Σ by T_Σ.

Building Trees

We can construct new trees, by joining given trees under a new root node. For a symbol $f \in \Sigma_n$ and trees $t_1, \ldots, t_n \in \mathrm{T}_\Sigma$ we write $f(t_1, \ldots, t_n)$ for the tree t with $\mathrm{pos}(t) = \{\varepsilon\} \cup \bigcup_{k=1}^{n} k\,\mathrm{pos}(t_k)$ and

$$t(x) = \begin{cases} f & \text{if } x = \varepsilon \\ t_i(y) & \text{if } x = iy \text{ for } i \in \mathbb{N} \text{ and } y \in \mathbb{N}^*. \end{cases}$$

A second construction of a new tree from an existing one is by selecting the *subtree* below a node. Let $t \in \mathrm{T}_\Sigma$ and $x \in \mathrm{pos}(t)$, we write $t|_x$ for the tree t' defined by $\mathrm{pos}(t') = \{y \in \mathbb{N}^* \mid xy \in \mathrm{pos}(t)\}$ and $t'(y) = t(xy)$.

Substitutions in Trees

Given trees $s, t \in \mathrm{T}_\Sigma$ and a position $x \in \mathrm{pos}(t)$ let the *substitution* $t[x \leftarrow s]$ be the tree obtained from t by replacing the subtree $t|_x$ at x in t by s. Formally, we define the tree $t[x \leftarrow s] = t'$ by

$$\mathrm{pos}(t') = (\mathrm{pos}(t) \setminus x\mathbb{N}^*) \cup x\,\mathrm{pos}(s)$$
$$t'(y) = \begin{cases} s(y') & \text{if } y = xy' \\ t(y) & \text{otherwise.} \end{cases}$$

When applying several substitutions in a row, the order of substitution may matter if the positions of later substitutions fall within subtrees which have been substituted before. This cannot happen if all positions are pairwise incomparable with respect to the prefix order.

For a sequence of positions $x_1, \ldots, x_n$ and trees $s_1, \ldots, s_n$ we define

$$t[x_i \leftarrow s_i]_{i=1,\ldots,n} = t[x_1 \leftarrow s_1][x_2 \leftarrow s_2] \cdots [x_n \leftarrow s_n].$$

If the x_i's are pairwise $\preceq$-incomparable, the order of the substitutions does not matter. For a $\preceq$-antichain $M \subseteq \mathrm{pos}(t)$, i.e., $x \not\preceq y \not\preceq x$ for all $x, y \in M$, let $t[M \leftarrow s] = t[x \leftarrow s]_{x \in M}$.

2.3.2 Tree Automata

For the rest of this section let Σ be a fixed ranked alphabet. A tree automaton is defined similar to a word automaton, but instead of considering two states and a label in the transition relation, a tree automaton considers the state at a node, the label of this node, and the states at all child nodes.

Definition 2.21. A *tree automaton* over Σ is a quadruple $A = (Q, T, I, F)$ where

1. Q is a finite, non-empty set – the set of states
2. $T \subseteq \bigcup_{n \geq 1} Q \times \Sigma_n \times Q^n$ – the transition relation
3. $I \subseteq Q$ – the set of initial states
4. $F \subseteq Q \times \Sigma_0$ – the acceptance condition.

Let t be any tree. A run of A on t is a mapping $\rho \colon \mathrm{pos}(t) \to Q$ that satisfies

$$\big(\rho(x), t(x), \rho(x1), \ldots, \rho(xn_x)\big) \in T$$

for all $x \in \mathrm{inner}(t)$, where $n_x = \mathit{arity}(t(x))$. A run ρ is successful if $\rho(\varepsilon) \in I$ and $(\rho(x), t(x)) \in F$ for all $x \in \mathrm{leaf}(t)$. The language $\mathrm{L}(A)$ of A is the set of all trees t such that there exists a successful run of A on t.

The automaton A is called *top-down deterministic* if $|I| \leq 1$ and $|\{\overline{q} \in Q^n \mid (p, f, \overline{q}) \in T\}| \leq 1$ for all $p \in Q$ and $f \in \Sigma_n$ with $n \geq 1$. We say A is *top-down complete* if $|I| \geq 1$ and $|\{\overline{q} \in Q^n \mid (p, f, \overline{q}) \in T\}| \geq 1$ for all $p \in Q$ and $f \in \Sigma_n$ with $n \geq 1$. If A is both, top-down deterministic and top-down complete, we can regard T as a function $\delta = \bigcup_{n \geq 1} \delta_n$ with $\delta_n \colon Q \times \Sigma_n \to Q^n$. We also write $A = (Q, \delta, q_0, F)$ for a top-down deterministic and complete tree automaton.

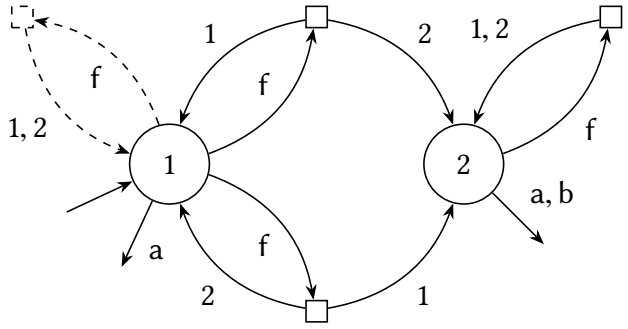


Figure 2.3: Automaton A from Example 2.22

Furthermore, A is *bottom-up deterministic* if $|\{q \in Q \mid (q, a) \in F\}| \leq 1$ for all $a \in \Sigma_0$ and $|\{q \in Q \mid (q, f, \overline{p}) \in T\}| \leq 1$ for all $f \in \Sigma_n$ and $\overline{p} \in Q^n$. The automaton A is *bottom-up complete* if $|\{q \in Q \mid (q, a) \in F\}| \geq 1$ for all $a \in \Sigma_0$ and $|\{q \in Q \mid (q, f, \overline{p}) \in T\}| \geq 1$ for all $f \in \Sigma_n$ and $\overline{p} \in Q^n$ If A is bottom-up deterministic and bottom-up complete, T and F represent a function $\delta = \bigcup_{n \geq 0} \delta_n$ where $\delta_n \colon \Sigma_n \times Q^n \to Q$ with $(\delta(a), a) \in F$ for all $a \in \Sigma_0$, and $(\delta(f, \overline{p}), f, \overline{p}) \in T$ for all $f \in \Sigma_n$ and $\overline{p} \in Q^n$. We also write (Q, δ, F') with $F' = I$ for a bottom-up deterministic and complete tree automaton.

A tree language $L \subseteq \mathrm{T}_\Sigma$ is called *recognizable* or *regular* if there is a tree automaton A with $\mathrm{L}(A) = L$.

Example 2.22. Let $\Sigma = \{\mathrm{f}, \mathrm{a}, \mathrm{b}\}$ with $\mathit{arity}((f)) = 2$ and $\mathit{arity}(\mathrm{a}) = \mathit{arity}(\mathrm{b}) = 0$. We consider the tree automaton A from Fig. 2.3. Ignore the dashed parts of the image for now. The picture is read as follows: circles represent states, whereas rectangles represent transitions. Arrows from states to transitions mean that this transition is applicable if the automaton is in the corresponding state and the symbol next to the arrow is at the current position. Arrows from rectangles to states say that if the transition of the rectangle is applicable, then the automaton transitions to the corresponding state at the i-th child, where i is the number next to the arrow. Single arrows into states denote initial states, whereas single arrow out of states tell that the state with the letter next to the arrow is accepting.

Figure 2.3 describes the automaton $A = (Q, T, I, F)$ with $Q = \{1, 2\}$, $T = \{(1, \mathrm{f}, 1, 2), (1, \mathrm{f}, 2, 1), (2, \mathrm{f}, 2, 2)\}$, $I = \{1\}$, and $F = \{(1, \mathrm{a}), (2, \mathrm{a}), (2, \mathrm{b})\}$. Note that though the representation of A as tuple has no inherent direction, the picture assumes a top-down approach. By reversing all arrows you obtain the same automaton, but viewed as a bottom-up automaton.

When in state 2, the automaton loops there and can exit at leaf nodes labelled with any letter. Hence, starting from state 2, all trees are accepted. In state 1 the automaton guesses non-deterministically whether to stay in the left or in the right

child in state 1. The other child enters state 2. Thus, the automaton guesses a path through the tree using state 1. At the end of this path, the automaton may only exit state 1 in an a labelled leaf node. Therefore, we obtain as language of A:

$$\mathrm{L}(A) = \left\{ t \in \mathrm{T}_\Sigma \;\middle|\; |\mathrm{pos}_{\mathrm{a}}(t)| \geq 1 \right\}.$$

Note that A is not top-down deterministic, since there are two transitions applicable in state 1 with f. In fact, this language is a standard example of a tree language which cannot be recognized by a top-down deterministic tree automaton. The automaton A is is bottom-up deterministic but not bottom-up complete, as there is no transition if both child states are in state 1 and a f is read. This can be fixed by adding the dashed transition to the automaton. This way, we obtain a bottom-up deterministic and complete automaton for $\mathrm{L}(A)$.

Having two notions of determinism for trees, one may ask if both deterministic automata models are equivalent to the general, non-deterministic model. It turns out that only the bottom-up deterministic model is as expressive as non-deterministic tree automata. These results are stated in the next two lemmas. For the proofs of this result, we refer the reader to [TATA], but also recall Example 2.22 for Lemma 2.24.

Lemma 2.23. Let A be a tree automaton. There is a bottom-up deterministic and bottom-up complete tree automaton A' such that $\mathrm{L}(A) = \mathrm{L}(A')$.

Lemma 2.24. There is an alphabet Σ and a regular tree language $L \subseteq \mathrm{T}_\Sigma$ such that L is not the language of any deterministic top-down tree automaton.

2.4 Probabilistic Automata on Trees

Probabilistic tree automata generalise top-down deterministic and complete tree automata by replacing the single unique tuple of children states by a distribution on all possible tuples of states. As we consider probabilistic tree automata only on finite trees, it is not necessary to employ measure theory in this case.

Definition 2.25. A *(top-down) probabilistic tree automaton* is a quadruple $A = (Q, \delta, \mu, F)$ where

1. Q is a finite, non-empty set – the set of states
2. $\delta = \bigcup_{n \geq 1} \delta_n$ with $\delta_n \colon Q \times \Sigma_n \to \Delta_0(Q^n)$ – the transition probability function
3. $\mu \in \Delta(Q)$ – the initial distribution

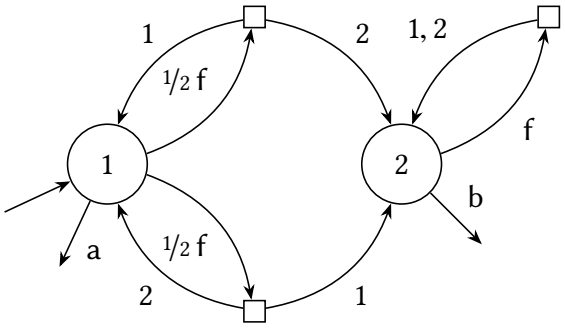


Figure 2.4: Automaton A' from Example 2.26

4. $F \subseteq Q \times \Sigma_0$ – the acceptance condition.

The behaviour $\|A\| \colon \mathrm{T}_\Sigma(V) \to [0,1]$ is given by

$$\|A\|(t) = \sum_{\substack{\rho \colon \mathrm{pos}(t) \to Q \\ \forall x \in \mathrm{leaf}(t) \colon (\rho(x), t(x)) \in F}} \mu(\rho(\varepsilon)) \prod_{x \in \mathrm{inner}(t)} \delta(\rho(x), t(x))(\rho(x1), \ldots, \rho(xn_x)),$$

for every $t \in \mathrm{T}_\Sigma$ where $n_x = \mathit{arity}(t(x))$.

We will make use of an alternative, recursive representation of $\|A\|$: we define functions $\delta_q \colon \mathrm{T}_\Sigma \to [0,1]$ for every $q \in Q$ by induction on the tree height. Let $t = f(t_1, \ldots, t_n)$. We set

$$\delta_q(t) = \begin{cases} \mathbb{1}_F(q, f) & \text{if } n = 0 \\ \sum_{q_1, \ldots, q_n \in Q} \delta(q, f)(q_1, \ldots, q_n) \prod_{i=1}^n \delta_{q_i}(t_i) & \text{if } n > 0. \end{cases}$$

Using induction one obtains for all $t \in \mathrm{T}_\Sigma$ that the following equation holds:

$$\|A\|(t) = \sum_{q \in Q} \mu(q)\, \delta_q(t).$$

Example 2.26. We return to Example 2.22. The automaton depicted in the corresponding picture Fig. 2.3 cannot be turned into a probabilistic automaton, as there are two applicable transitions in state 1 with the same letter f. Instead, we change this non-deterministic choice into a probabilistic one by chosen each of the two transitions with probability 1/2. The resulting probabilistic tree automaton A' is shown in Fig. 2.4. Probability values are written in front of the corresponding label, a missing number means probability 1.

In Example 2.22, we argued that the non-deterministic choice guesses a path in the tree, which has to end in an a labelled node. Now, each direction is chosen with probability 1/2. Thus, a leaf node at position $x \in \mathrm{pos}(t)$ of a tree t is reached with probability $(1/2)^{|x|}$. As this applies to every leaf node in the tree we obtain

$$\|A\|(t) = \sum_{x \in \mathrm{pos}_{\mathrm{a}}(t)} \left(\frac{1}{2}\right)^{|x|}.$$

Part I

Probabilistic Nivat-Theorem and Probabilistic MSO Logic

Chapter 3

Probabilistic Nivat Classes

In this chapter, we derive a probabilistic version of Nivat's theorem for finite and infinite words, as well as for finite trees. This result will allow us to characterise the recognizable probabilistic word series and tree series by recognizable languages, operations like homomorphic image and preimage, and application of a simple probability measure.

In Section 3.1 we recall the statement of the classical theorem. Afterwards, we introduce Bernoulli measures in Section 3.2 as measures that arise from sequences of unfair coin tosses. In Sections 3.3 and 3.4 we give our probabilistic variant of Nivat's theorems for words and finite trees, respectively.

The results on words have been published in [W12] and the results on finite trees in [W15].

3.1 Classical Nivat-theorem

Nivat's theorem, published in 1968 [N68], decomposes a rational transduction into a regular language and applications of homomorphisms and inverse homomorphisms. Thus, we will first introduce rational transducers. The following definitions and results, and an in-depth introduction to the topic can be found in [MS97].

Definition 3.1. Let Σ and Δ be two alphabets. A *rational transducer* from Σ^* to Δ^* is a quadruple $R = (Q, T, I, F)$ where

1. Q is a finite, non-empty set – the set of states,
2. $T \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times Q$ – the set of transitions,
3. $I \subseteq Q$ – the set of initial states,
4. $F \subseteq Q$ – the set of final states.

A *configuration* of R is a triple $(q, u, v) \in Q \times \Sigma^* \times \Delta^*$. We define a relation $\to$ on the configurations by $(p, au, v) \to (q, u, vb)$ if and only if $(p, a, b, q) \in T$. Using this relation, we define the transduction of R as function $\|R\| \colon \Sigma^* \to \mathcal{P}(\Delta^*)$ by

$$\|R\|(u) = \left\{ v \in \Delta^* \;\middle|\; \exists p \in I \colon \exists q \in F \colon (p, u, \varepsilon) \to (q, \varepsilon, v) \right\}.$$

The transduction $\|R\|$ can also be lifted to languages: let $L \subseteq \Sigma^*$ we define $\|R\|(L) = \bigcup_{u \in L} \|R\|(u)$.

Having defined rational transducers, we can now state Nivat's theorem.

Theorem 3.2 (Nivat's theorem). Let Σ and Δ be two alphabets, and $T \colon \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Delta^*)$ be any function. There is a rational transducer R from Σ^* to Δ^* with $\|R\| = T$ if and only if there exist an alphabet Γ, a regular language $L \subseteq \Gamma^*$, and homomorphisms $h \colon \Gamma^* \to \Sigma^*$ and $g \colon \Gamma^* \to \Delta^*$ such that

$$T(X) = g(h^{-1}(X) \cap L),$$

for all $X \subseteq \Sigma^*$.

One remarkable application of Nivat's theorem is the closure of cones under rational transductions. Let us first introduce cones. A *family of languages* is a collection $\mathcal{L}$ of formal languages of finite words such that $\mathcal{L}$ contains at least one non-empty language. Each $L \in \mathcal{L}$ is a language over some finite alphabet, but the alphabets do not need to be the same for two languages from $\mathcal{L}$. A *cone* is a family of languages, that is closed under homomorphic images, homomorphic preimages and intersection with regular languages. For example the family of regular languages and the family of context-free languages are cones. As immediate consequence of Nivat's theorem one obtains the following result.

Corollary 3.3. Let $\mathcal{L}$ be a cone. Then, $\mathcal{L}$ is closed under rational transductions.

3.2 Bernoulli Measures on Words and Trees

Consider an experiment like tossing an unfair coin. There are two possible outcomes: one with probability p and the other with probability $1-p$. Such experiments are called Bernoulli trials. A Bernoulli process consists of finite or infinitely many independent Bernoulli trials all with the same probability p. Thus, a Bernoulli process can be seen as tossing the same unfair coin many times in a row independently from each other.

Instead of considering only a binary outcome, one may be interested in any finite number of outcomes. So, instead of tossing an unfair coin, one could also

roll an unfair die. Thus, there is a finite set M of outcomes. Every $m \in M$ occurs with probability p_m and the mapping $m \mapsto p_m$ is a distribution on M. A finite or infinite sequence of such experiments is called a Bernoulli scheme. Whereas finite repetition of these trials results in a finite probability space, an infinite number of experiments yields an uncountable probability space.

Note that the length of a trial or scheme is fixed in advance. Thus, we only obtain a measure on sequences of the same length.

Definition 3.4. Let M be a finite set and $d \in \Delta(M)$ be a distribution on M.

1. For every $n \in \mathbb{N}$, we define a measure B_d^n on $\mathcal{P}(M^n)$ by
$$\mathrm{B}_d^n(\{m_1 \cdots m_n\}) = \prod_{i=1}^{n} d(m_i),$$
for all $m_1, \ldots, m_n \in M$.
2. We define a measure B_d^ω on $\mathcal{B}(M^\omega)$ by
$$\mathrm{B}_d^\omega(m_1 \cdots m_k M^\omega) = \prod_{i=1}^{n} d(m_i),$$
for all $m_1, \ldots, m_k \in M$ and $k \geq 0$.

In both cases we call B_d^n the *Bernoulli measure* of d on M^n for $n \in \mathbb{N} \cup \{\omega\}$.

For any probability value p, we write B_p^n for the Bernoulli measure of d_p on $\{0,1\}^n$, where $d_p(1) = p$ and $d_p(0) = 1 - p$. Moreover, we write B_d (B_p) for B_d^n (B_p^n) if n is understood.

For an application to trees, considering a linear sequence of outcomes is not sufficient. Instead, we extend the Bernoulli measure to arbitrary domains. As we only consider finite trees here, we only give the definition for finite sets. Nevertheless, an extension to countable domains is easily possible using techniques similar to Section 5.1.

Definition 3.5. Let D be a finite, non-empty domain, M a finite, non-empty set, and p a distribution on M. We define a distribution B_d^D on the finite set M^D by
$$\mathrm{B}_d^D(u) = \prod_{x \in D} d(u(x)),$$
for all $u \in M^D$. As before, we let B_p^D for $p \in [0,1]$ be the distribution $\mathrm{B}_{d_p}^D$ on $\{0,1\}^D$ where $d_p(1) = p$ and $d_p(0) = 1 - p$. If D is understood, we just write B_d for B_d^D.

Note that we gave a definition of B_d^D as distribution $M^D \to [0,1]$, but recall that this definition unique extends to a measure on $(M^D, \mathcal{P}(M^D))$, as M^D is finite. We will also write B_p^D for this measure.

3.3 Nivat classes for words

Before we give the probabilistic Nivat theorem, we recall some notation. Given a homomorphism $h\colon \Gamma^\infty \to \Sigma^\infty$, we write $h(L)$ for the image of a language $L \subseteq \Gamma^\infty$ under h, and $h^{-1}(K)$ for the preimage of $K \subseteq \Sigma^\infty$. Moreover, for a measure μ on $\mathcal{B}(\Sigma^\infty)$ and a set $L \subset \Gamma^\infty$ such that $h(L)$ is measurable, we write $(\mu \circ h)(L)$ for $\mu(h(L))$. Note that $\mu \circ h$ is not a measure. Recall that a homomorphism $h\colon \Sigma^\infty \to \Gamma^\infty$ is a relabelling if $h(a) \in \Gamma$ for all $a \in \Sigma$.

Definition 3.6. Let Σ be a finite alphabet. The *Nivat-class* $\mathcal{N}(\Sigma^\infty)$ consists of all functions $S\colon \Sigma^\infty \to [0,1]$ such that there are

1. a finite alphabet Γ and a finite, non-empty set M,
2. a regular language $L \subseteq \Gamma^\infty$,
3. relabellings $h\colon \Gamma^\infty \to \Sigma^\infty$ and $g\colon \Gamma^\infty \to M^\infty$,
4. a distribution d on M,

with

$$S(w) = (\mathrm{B}_d \circ g)\big(h^{-1}(\{w\}) \cap L\big), \tag{3.1}$$

for all $w \in \Sigma^\infty$.

In the simple case that $\Gamma = \Sigma \times M$, and the functions h and g are the canonical projections, (3.1) can be written as

$$S(w) = \mathrm{B}_d(\{u \in M^\omega \mid (w,u) \in L\}),$$

where we use tuple notation for words over tuples: let $w \in \Sigma^\infty$ and $u \in M^\infty$ with $|w| = |u|$. We consider the tuple (w,u) as word v over $(\Sigma \times M)^\infty$ where $|v| = |w|$ and $v_i = (w_i, u_i)_{i \in \mathrm{pos}(v)}$ for $w = (w_i)_{i \in \mathrm{pos}(u)}$ and $u = (u_i)_{i \in \mathrm{pos}(v)}$.

Before we come to the main result of this section – the equivalence of Nivat classes and recognizable functions – we still need to show that the definition is sound, i.e., that the set $g(h^{-1}(\{w\}) \cap L)$ is measurable. As relabellings are essentially projections and regular languages of words are Borel sets, any set of the form $g(h^{-1}(\{w\}) \cap L)$ is an analytic set and therefore universally measurable, i.e., measurable in every complete probability space. We show in the next lemma, that these sets are even Borel sets. This will be a consequence from the fact that every regular language is a Borel set.

Lemma 3.7. Let $L \subseteq \Sigma^\infty$ be a regular tree language. Then, L is also measurable, i.e, $L \in \mathcal{B}(\Sigma^\infty)$.

Proof. Let $A = (Q, \delta, q_\iota, F, \mathcal{R})$ be a deterministic and complete Muller-automaton with $\mathrm{L}(A) = L$. Furthermore, let $\mathfrak{S} = \{\rho \in Q^\omega \mid \inf(\rho) \in \mathcal{R}\}$ the set of successful runs on an infinite word. As seen after Definition 2.17, we already know that $\mathfrak{S}$ is measurable in $\mathcal{B}(Q^\omega)$.

Let $r\colon \Sigma^\omega \to Q^\omega$ map every word w to its unique run in A. Since the preimage of a cylinder set $p_0 \cdots p_n Q^\omega$ with $p_0 = q_\iota$ under r is given by

$$r^{-1}(p_0 \cdots p_n Q^\omega) = \bigcup_{\substack{w_1,\cdots,w_n \in \Sigma \\ \delta(p_{i-1},w_i)=p_i \text{ for all } i=1,\ldots,n}} w_1 \cdots w_n \Sigma^\omega,$$

we conclude that r is a continuous function. Therefore, $r^{-1}(\mathfrak{S})$ is a Borel set. As $L = r^{-1}(\mathfrak{S}) \cup \bigcup_{w \in L \cap \Sigma^*} \{w\}$, the proof is complete. ∎

Corollary 3.8. Let $L \subseteq \Gamma^\infty$ be a recognizable language, $h\colon \Gamma^\infty \to \Sigma^\infty$ and $g\colon \Gamma^\infty \to M^\infty$ be relabellings, and $w \in \Sigma^\infty$. Then, $g(h^{-1}(\{w\}) \cap L) \in \mathcal{B}(M^\infty)$.

Proof. Let $\pi\colon \Gamma^\infty \to (\Sigma \times M)^\infty$ be the homomorphism with $\pi(v) = (h(v), g(v))$. As the regular languages are closed under homomorphic image, the language $L' = \pi(L \cap \Gamma^\omega)$ is again regular. By Lemma 3.7 we also know that $L' \in \mathcal{B}((\Sigma \times M)^\infty)$. Fix $w = w_1 w_2 \cdots \in \Sigma^\omega$ and consider the function $\kappa_w\colon M^\omega \to (\Sigma \times M)^\omega$ given by $\kappa_w(u) = (w, u)$ where $(w, u) = (w_1, u_1)(w_2, u_2) \cdots$ and $u = u_1 u_2 \cdots$. Clearly, this function is continuous and hence measurable. Thus, $\kappa_w^{-1}(L') \in \mathcal{B}(\Sigma^\omega)$. Moreover, we obtain

$$\begin{aligned} \kappa_w^{-1}(L') &= \left\{ u \in M^\omega \;\middle|\; (w, u) \in L' \right\} \\ &= \left\{ g(v) \;\middle|\; \exists v \in \Gamma^\omega \colon h(v) = w \right\} \\ &= g\big(h^{-1}(\{w\}) \cap (L \cap \Gamma^\omega)\big). \end{aligned}$$

As the image of finite and infinite words under g are disjoint, we obtain $g(h^{-1}(\{w\}) \cap L) = g(h^{-1}(w) \cap L \cap \Gamma^\omega) \cup g(h^{-1}(\{w\}) \cap L \cap \Gamma^*)$ which is measurable, as the second term is at most countable. ∎

After having shown that the terms in Definition 3.6 are well-defined, we show a probabilistic version of Theorem 3.2. This statement resembles the classical result, only introducing an additional measure. Note that the empty word is handled differently in probabilistic automata and Nivat classes: as B_d^0 is a probability measure on $(\{\varepsilon\}, \{\emptyset, \{\varepsilon\}\})$, the only possible outcomes for the empty word are 0 and 1. On the other hand, probabilistic automata can assign any value to the probability of ε. Therefore, we explicitly set the value of ε to 0 for Nivat classes in the next theorem.

Theorem 3.9. Let Σ be an alphabet and $S\colon \Sigma^\infty \to [0,1]$ be a probabilistic series. The following statements are equivalent:

1. $S = \|A\|$ for some probabilistic Muller-automaton A,
2. $S_+ \in \mathcal{N}(\Sigma)$,

where $S_+(w) = S(w)$ for all $w \neq \varepsilon$ and $S_+(\varepsilon) = 0$. The translations are effective in both directions.

Before we can prove Theorem 3.9, we need two axillary results: In Proposition 3.10 we give a construction to embed several finite distributions into a single one. Proposition 3.11 shows the correctness of an automata construction we will use in two places in the proof of Theorem 3.9.

Proposition 3.10. Let $M_1, \dots, M_n$ be finite sets and d_i be a distribution on M_i for every $i = 1, \dots, n$. There are a finite set M, a distribution d on M, and functions $\pi_i\colon M \to M_i$ such that $d_i(X) = d(\pi_i^{-1}(X))$ for every $X \subseteq M_i$ and $i = 1, \dots, n$.

Moreover, the size of M is bounded by $\sum_{i=1}^n |M_i|$.

A simple construction with the above properties, except the size constraints, would be defining $(M, d) = \bigotimes_{i=1}^n (M_i, d_i)$, but the size of M would be $\prod_{i=1}^n |M_i|$ which would cause an exponential blowup in upcoming constructions. Therefore, we present a different construction with only polynomial blowup.

Proof (of Proposition 3.10). We may assume $M_i = \{1, \dots, m_i\}$ with $m_i \geq 1$ for every $i = 1, \dots, n$. Let $V_i = \{d_i(\{1, \dots, k\}) \mid 1 \leq k \leq m_i\} \cup \{0, 1\}$ and $V = \bigcup_{i=1}^n V_i$. Thus, $|V_i| \leq \sum_{i=1}^n m_i$. Let $\{v_1, \dots, v_\ell\} = V$ be an enumeration of V with $v_i < v_j$ for all $1 \leq i < j \leq \ell$. We define $M = \{1, \dots, \ell\}$ and $d(\{i\}) = v_i - v_{i-1}$ where we set $v_0 = 0$. As $v_\ell = 1$, d is a distribution on M. We define the functions $\pi_i\colon M \to M_i$ by

$$\pi_i(m) = \min\bigl\{k \in M_i \bigm| v_m \leq d_i(\{1, \dots, k\})\bigr\}.$$

We show $d_i = d \circ \pi_i^{-1}$. Let $k \in M_i$. By definition of π_i, we have $\pi_i^{-1}(\{k\}) = \{m \in M \mid d_i(\{1, \dots, k-1\}) < v_m \leq d_i(\{1, \dots, k\})\}$. By definition of V there are a $m_-, m_+ \in M \cup \{0\}$ with $v_{m_-} = d_i(\{1, \dots, k-1\})$ and $v_{m_+} = d_i(\{1, \dots, k\})$. Thus, $d(\pi_i^{-1}(\{k\})) = \sum_{m=m_-+1}^{m_+} (v_m - v_{m-1}) = v_{m_+} - v_{m_-} = d_i(\{1, \dots, k\}) - d_i(\{1, \dots, k-1\}) = d_i(\{k\})$. This completes the proof. ∎

Proposition 3.11. Let Σ and M be alphabets and d a distribution on M. Let further $A' = (Q, T, \{q_\iota\}, F, \mathcal{R})$ be a deterministic and complete Muller-automaton over $\Sigma \times M$. The probabilistic Muller-automaton A given by $A = (Q, \delta, \mathbb{1}_{\{q_\iota\}}, F, \mathcal{R})$ and $\delta(p, a, q) = d(\{m \in M \mid (p, (a, m), q) \in T\}$ satisfies $\|A\|(w) = \mathrm{B}_d(\{u \in M^{\mathrm{pos}(w)} \mid (w, u) \in \mathrm{L}(A')\})$ for all $w \in \Sigma^\infty$.

Proof. For a word $w \in \Sigma^\infty$ we define a function $\kappa_w \colon M^\infty \to Q^\infty$ mapping a word $u \in M^\infty$ to the unique run of (w, u) in A'. Fix a $w \in \Sigma^\infty$. We show $\Pr_A^w = \mathrm{B}_d \circ \kappa_w^{-1}$. Let $q_0 \cdots q_n \in Q^*$ with $n \leq |w|$. We obtain

$$\begin{aligned}\Pr_A^w(q_0 \cdots q_n Q^\infty) &= \mu(q_0) \prod_{i=1}^{n} \delta(q_{i-1}, w_i, q_i) \\ &= \mathbb{1}_{\{q_\iota\}} \prod_{i=1}^{n} d(\{m \in M \mid (q_{i-1}, (w_i, m), q_i) \in T\})\end{aligned}$$

By use of distributivity we obtain

$$\begin{aligned}&= \mathbb{1}_{\{q_\iota\}} \sum_{\substack{m_1,\ldots,m_n \in M \\ \forall i \colon (q_{i-1},(w_i,m_i),q_i) \in T}} \prod_{i=1}^{n} d(m_i) \\ &= \mathbb{1}_{\{q_\iota\}} \sum_{\substack{m_1,\ldots,m_n \in M \\ \forall i \colon (q_{i-1},(w_i,m_i),q_i) \in T}} \mathrm{B}_d(m_1 \cdots m_n M^\infty)\end{aligned}$$

Note that $\kappa_w(m_1 m_2 \cdots) = r_0 r_1 \cdots$ with $r_0 = q_\iota$ and $(r_{i-1}, (w_i, m_i), r_i) \in T$ for all $i \geq 1$. Furthermore, a run $\rho = r_0 r_1 \cdots$ starts with $q_0 \cdots q_n$ if and only if $\rho \in q_0 \cdots q_n Q^\infty$. We conclude

$$\begin{aligned}&= \mathrm{B}_d(\{m_1 m_2 \cdots \mid \kappa_w(m_1 m_2 \cdots) \in q_0 \cdots q_n Q^\omega\}) \\ &= (\mathrm{B}_d \circ \kappa_w^{-1})(q_0 \cdots q_n Q^\infty).\end{aligned}$$

Thus, $\Pr_A^w = \mathrm{B}_d \circ \kappa_w^{-1}$. Let $\mathfrak{S} \subseteq Q^\infty$ be the set of accepting runs. We conclude $\|A\|(w) = \Pr_A^w(\mathfrak{S}) = \mathrm{B}_d(\kappa_w^{-1}(\mathfrak{S}))$. Note that $\kappa_w^{-1}(\mathfrak{S})$ contains exactly the words $u \in M^\infty$ with $(w, u) \in \mathrm{L}(A')$. This completes the proof. ∎

We are now ready to prove the equivalence of probabilistic automata and probabilistic Nivat classes for words.

Proof (of Theorem 3.9). Let S be recognizable by some probabilistic Muller-automaton. Using the standard construction one obtains a probabilistic Muller-automaton with unique initial state and $\|A\| = S_+$: given a probabilistic Muller-automaton $A = (Q, \delta, \mu, F, \mathcal{R})$ recognizing S, one defines a new automaton $A' = (Q \cup \{q_\iota\}, \delta', \mu', F, \mathcal{R})$ with $\mu'(q_\iota) = 1$, $\mu(q) = 0$, $\delta'(q_\iota, a, q) = \sum_{p \in Q} \mu(p)\, \delta(p, a, q)$, $\delta'(p, a, q_\iota) = 0$, and $\delta'(q_\iota, a, q_\iota) = 0$ for all $p, q \in Q$ and $a \in \Sigma$. Then, $\|A'\| = S_+$ holds. Thus, we can assume a probabilistic Muller-automaton $A = (Q, \delta, \mu, F, \mathcal{R})$ with $\mu(q_\iota) = 1$ for some $q_\iota \in Q$, and $\|A\| = S_+$.

By Proposition 3.10, there is a finite set M, a distribution d on M and for every $(p,a) \in Q \times \Sigma$ a function $\pi_{p,a} \colon M \to Q$ with $\delta(p,a)(X) = d(\pi_{p,a}^{-1}(X))$ for all $X \subseteq Q$. Let $\Gamma = \Sigma \times M$, and $h \colon \Gamma^\infty \to \Sigma^\infty$ and $g \colon \Gamma^\infty \to M^\infty$ the canonical projections. We define a deterministic and complete Muller-automaton $A' = (Q, T, q_\iota, F, \mathcal{R})$ by $T = \{(p,(a,m),\pi_{p,a}(m)) \mid p \in Q, (a,m) \in \Gamma\}$. This definition implies $\delta(p,a,q) = d(\pi_{p,a}^{-1}(\{q\})) = d(\{m \in M \mid (p,(a,m),q) \in T\})$. Thus, by Proposition 3.11, we obtain $\|A\|(w) = \mathrm{B}_d(\{u \in M^{\mathrm{pos}(w)} \mid (w,u) \in \mathrm{L}(A')\})$ which is just (3.1). Thus, $S_+ \in \mathcal{N}(\Sigma)$.

Conversely, assume L, M, d, Γ, h, and g are given as in Definition 3.6 such that $S_+(w) = (\mathrm{B}_d \circ g)(h^{-1}(w) \cap L)$ for all $w \in \Sigma^\infty$. Let $\kappa \colon \Gamma \to \Sigma \times M$ be given by $\kappa(a) = (h(a), g(a))$. Then, κ extends uniquely to a homomorphism $\kappa \colon \Gamma^\infty \to (\Sigma \times M)^\infty$ and the language $L' = \kappa(L)$ is again regular. Moreover, we have $g = \pi_2 \circ \kappa$ and $h = \pi_1 \circ \kappa$, where π_i is the projection on the i-th component. Let $A = (Q, T, q_\iota, F, \mathcal{R})$ be a deterministic and complete Muller-automaton with $\mathrm{L}(A) = L'$. We construct a probabilistic Muller automaton A' over Σ by letting $A' = (Q, \delta, \mathbb{1}_{\{q_\iota\}}, F, \mathcal{R})$ with $\delta(p,a,q) = d(\{m \in M \mid (p,(a,m),q) \in T\})$. We obtain the following:

$$\begin{aligned} S_+(w) &= (\mathrm{B}_d \circ g)(h^{-1}(\{w\}) \cap L) \\ &= \mathrm{B}_d(\pi_2(\kappa(\kappa^{-1}(\pi_1^{-1}(\{w\})) \cap L))) \end{aligned}$$

Using the general identity $f(f^{-1}(M) \cap N) = M \cap f(N)$ for all functions $f \colon X \to Y$ and sets $M \subseteq Y$ and $N \subseteq X$:

$$\begin{aligned} &= (\mathrm{B}_d \circ \pi_2)\big(\pi_1^{-1}(\{w\}) \cap \kappa(L)\big) \\ &= \mathrm{B}_d(\{u \in M^\infty \mid (w,u) \in L'\}) \end{aligned}$$

By Proposition 3.11 this is just the behaviour of A':

$$= \|A'\|(w).$$

Thus, $\|A'\| = S_+$. We still need to extend A' to recognize S and not S_+. Let $\lambda = S(\varepsilon)$. We define $A_1 = (Q \cup \{q_\iota, q_f\}, \delta_1, \mu_1, F \cup \{q_\iota\})$ where $\mu_1(q_\iota) = \lambda$, $\mu_1(q_f) = 1 - \lambda$, and $\mu_1(q) = 0$ for all $q \in Q$. Furthermore, let $\delta_1(p,a,q) = \delta(p,a,q)$, $\delta(q,a,r) = 0$, and $\delta(r,a,q) = \sum_{p \in Q} \mu'(p)\,\delta(p,a,q)$ for $r \in \{q_\iota, q_f\}$ and all $p, q \in Q$. By construction, after reading at least one letter, the probability to reach a state $q \in Q$ in A' is the same as the probability to reach the same state q in A_1. By choice of μ_1 we additionally have $\|A_1\|(\varepsilon) = \lambda = S(\varepsilon)$. Therefore, we obtain $\|A_1\| = S$. ∎

3.4 Nivat Classes for Trees

After having given the definition of probabilistic Nivat classes for words and having shown their equivalence to the recognizable word series, we next look at finite trees and transfer these results on words to trees.

It turns out that there is a difference for trees, whether the regular language, which occurs in the definition of Nivat class, is recognizable by a top-down deterministic tree automaton or not. This is different from the word case, as every regular word language admits a deterministic word automaton that recognizes it.

A *relabelling* between to rank alphabets Σ and Γ is a function $h\colon \Sigma \to \Gamma$ with $arity_\Sigma(a) = arity_\Gamma(h(a))$ for all $a \in \Sigma$. Then h extends to a function $h\colon \mathrm{T}_\Sigma \to \mathrm{T}_\Gamma$ by $\mathrm{pos}(h(t)) = \mathrm{pos}(t)$ and $h(t)(x) = h(t(x))$ for all $x \in \mathrm{pos}(t)$ and $t \in \mathrm{T}_\Sigma$.

We also use functions mapping tree labels to an arbitrary set M. A function $g\colon \Sigma \to M$ extends to a function $g\colon \mathrm{T}_\Sigma \to \bigcup_{D\colon D \text{ tree domain}} M^D$ by setting $g(t) = \tau$ where $\tau\colon \mathrm{pos}(t) \to M$ with $\tau(x) = g(t(x))$ for all $x \in \mathrm{pos}(t)$ and $t \in \mathrm{T}_\Sigma$. Note that, $g(t)$ is not a ranked tree though. As before, we write $\mathrm{B}_d^D \circ g$ for the function $t \mapsto \mathrm{B}_d^D(g(t))$.

Definition 3.12. Let Σ be a rank alphabet. The *Nivat-class* $\mathcal{N}(\mathrm{T}_\Sigma)$ consists of all tree series $S\colon \mathrm{T}_\Sigma \to [0,1]$ such that there are

1. a rank alphabet Γ and a finite, non-empty set M,
2. a regular tree language $L \subseteq \mathrm{T}_\Gamma$,
3. relabellings $h\colon \Gamma \to \Sigma$ and $g\colon \Gamma \to M$,
4. a distribution d on M,

such that for all $t \in \mathrm{T}_\Sigma$ we have

$$S(t) = (B_d \circ g)\big(h^{-1}(\{t\}) \cap L\big). \tag{3.2}$$

The *deterministic Nivat-class* $\mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma)$ comprises all tree series S such that conditions 1. – 4. are satisfied, Eq. (3.2) holds, and additionally:

5. L is recognizable by a deterministic top-down tree automaton,
6. the mapping $\Gamma \to \Sigma \times M$ given by $a \mapsto (h(a), g(a))$ is injective.

Similar to the word case we consider the simple case that Γ is the ranked alphabet $\Sigma \times M$ with $arity_\Gamma((f,m)) = arity_\Sigma(f)$, and the functions h and g are the canonical projections. Then, (3.2) can be written as

$$S(t) = \mathrm{B}_d(\{u \in M^{\mathrm{pos}(t)} \mid (t,u) \in L\}),$$

where we use tuple notation for tree of tuples: let $t \in \mathrm{T}_\Sigma$ and $u \in M^{\mathrm{pos}(t)}$. We consider the tuple (t,u) as tree t' over $\Sigma \times M$ where $\mathrm{pos}(t) = \mathrm{pos}(t')$ and $t'(x) = (t(x), u(x))$ for all $x \in \mathrm{pos}(t)$.

By definition, $\mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma) \subseteq \mathcal{N}(\mathrm{T}_\Sigma)$ holds. We show that this inclusion is strict.

Lemma 3.13. Let Σ be a rank alphabet with at least one symbol of arity at least 2, and at least one leaf symbol. Then, there is a tree series $S \in \mathcal{N}(\mathrm{T}_\Sigma) \setminus \mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma)$.

If Σ only contains unary symbols and leaf symbols, tree languages over Σ are effectively word languages, and $\mathcal{N}(\Sigma) = \mathcal{N}_\mathrm{D}(\Sigma)$. This can be seen from the proof of Theorem 3.9.

Proof. The argument is similar to the argument why deterministic top-down automata do not recognize all regular tree languages. Let $f, a \in \Sigma$ with $\mathit{arity}(f) \geq 2$ and $\mathit{arity}(a) = 0$. We consider the two trees $t_1 = f(a, \ldots, a, f(a, \ldots, a))$ and $t_2 = f(f(a, \ldots, a), a, \ldots, a)$. Let $S = \mathbb{1}_{\{t_1, t_2\}}$ and assume $S \in \mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma)$. Let Γ, M, h, g, L as in Definition 3.12. As $S(t_1) > 0$, there is a tree

$$s_1 = u\big(v_1\big(w_1, \ldots, w_n\big), v_2, \ldots, v_n\big) \in \mathrm{T}_\Gamma$$

with $s_1 \in L$, $h(s_1) = t_1$, and $d(g(u)) > 0$, $d(g(v_i)) > 0$, and $d(g(w_i)) > 0$ for all $i = 1, \ldots, n$. Moreover, as $S(t_2) = 1$, there is a tree

$$s_2 = u\big(v_1', \ldots, v_{n-1}', v_n'\big(w_1', \ldots, w_n'\big)\big) \in \mathrm{T}_\Gamma,$$

with $s_2 \in L$, $h(s_2) = t_2$, and $d(g(v_i')) > 0$, and $d(g(w_i')) > 0$ for all $i = 1, \ldots, n$. Note that the root symbol of s_1 and s_2 is the same. We can choose the same root symbol for s_2, as otherwise $S(t_2) < 1$. Since L is top-down deterministic recognizable, L has the subtree exchange property (see for example [MNS08]). Thus, the tree s given by

$$s = (f, u)\big((a, v_1'), (a, v_2), \ldots, (a, v_n)\big)$$

is also contained in L. This implies $S(t) > 0$ for $t = f(a, \ldots, a)$. A contradiction to the definition of S. ∎

3.4.1 Nivat Classes and Probabilistic Tree Automata

We give the relation of Nivat classes for trees to probabilistic tree automata. There is a connection between deterministic Nivat classes and probabilistic tree automata: both use the top-down model. Thus, it is not surprising that probabilistic tree automata correspond to the deterministic Nivat class and not to the full class.

Theorem 3.14. Let $S \colon \mathrm{T}_\Sigma \to [0, 1]$ a tree series. The following statements are equivalent:

1. $S = \|A\|$ for a top-down probabilistic tree automaton A,
2. $S \in \mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma)$.

The translations are effective in both directions.

The proof of this statement involves a technical difficulty: whereas the Bernoulli distribution in the definition of $\mathcal{N}(\mathrm{T}_\Sigma)$ assigns a probability value to every position in a tree, probabilistic tree automata employ a probability distribution only for the root node and for the inner nodes, but not for leaf nodes. To overcome this difference, we use probabilistic tree automata with additional final weights. This model allows us to assign a probability to leaf nodes. Nevertheless, PTA with final weights are not more expressive than standard probabilistic tree automata.

Definition 3.15. A *probabilistic tree automaton with final weights* is a quadruple $A = (Q, \delta, \mu, \gamma)$ where Q, δ, μ are defined as in Definition 2.25 and $\gamma\colon Q \times \Sigma_0 \to [0,1]$ is the final weight function.

The behaviour of A is the function $\|A\|\colon \mathrm{T}_\Sigma \to [0,1]$ given by

$$\|A\|(t) = \sum_{\rho\colon \mathrm{pos}(t)\to Q} \mu(\rho(\varepsilon)) \left(\prod_{x \in \mathrm{inner}(t)} \delta(\rho(x), t(x))(\rho(x1), \ldots, \rho(xn_x)) \right) \cdot \prod_{x \in \mathrm{leaf}(t)} \gamma(\rho(x), t(x)),$$

where $n_x = \mathit{arity}(t(x))$.

The behaviour of a PTA with final weights can also be written using induction on the height of the input tree. For a PTA with final weights $A = (Q, \delta, \mu, \gamma)$ we set

$$\delta_q(a) = \gamma(q, a)$$
$$\delta_q(f(t_1, \ldots, t_n)) = \sum_{q_1, \ldots, q_n \in Q} \delta(q, f)(q_1, \ldots, q_n) \prod_{i=1}^{n} \delta_{q_i}(t_i),$$

for $a \in \Sigma_0$, $f \in \Sigma_n$, $t_1, \ldots, t_n \in \mathrm{T}_\Sigma$, and $n > 0$. Using these definitions the behaviour of A is given by $\|A\| = \sum_{q \in Q} \mu(q)\, \delta_q$.

Next, we show that final weights do not add expressive power to probabilistic tree automata.

Lemma 3.16. Probabilistic tree automata and probabilistic tree automata with final weights are equally expressive.

Proof. The direction from PTA to PTA with final weights is straightforward: given a PTA $A = (Q, \delta, \mu, F)$ we define the PTA with final weights $A' = (Q, \delta, \mu, \mathbb{1}_F)$. This automaton recognizes the same tree series.

Conversely, let $A = (Q, \delta, \mu, \gamma)$ be a PTA with final weights. We define a probabilistic tree automaton which probabilistically chooses an acceptance condition in every step and verifies in leaf nodes that the chosen condition is actually satisfied. Let the probability distribution d_γ on $\mathcal{P}(Q \times \Sigma_0)$ be given by

$$d_\gamma(M) = \prod_{(q,a)\in M} \gamma(q,a) \prod_{(q,a)\in Q\times\Sigma_0\setminus M} (1-\gamma(q,a)).$$

This distribution satisfies $d_\gamma(\{P \subseteq Q \times \Sigma_0 \mid (q,a) \in P\}) = \gamma(q,a)$ for all $(q,a) \in Q \times \Sigma_0$. Let A' be the PTA $A' = (Q', \delta', \mu', F')$ where

$$Q' = Q \times \mathcal{P}(Q \times \Sigma_0), \qquad \mu'(p,P) = \mu(p)\, d_\gamma(P), \qquad F' = \big\{((p,P),a) \,\big|\, (p,a) \in P\big\},$$

$$\delta'((p,P),f)((r_1,R_1),\dots,(r_n,R_n)) = \delta(p,f)(r_1,\dots,r_n) \prod_{i=1}^{n} d_\gamma(R_i).$$

We show $\sum_{P\subseteq Q\times\Sigma_0} d_\gamma(P)\, \delta'_{(p,P)}(t) = \delta_p(t)$. First consider a tree $a \in \Sigma_0$ of height 0. We obtain

$$\begin{aligned}\sum_{P\subseteq Q\times\Sigma_0} d_\gamma(P)\, \delta'_{(p,P)}(a) &= \sum_{P\subseteq Q\times\Sigma_0} d_\gamma(P)\, \mathbb{1}_P(p,a) \\ &= d_\gamma(\{P \mid (p,a) \in P\}) = \gamma(p,a) = \delta_p(a).\end{aligned}$$

Next, we consider trees of height at least 1. Let $t = f(t_1,\dots,t_n)$ with $f \in \Sigma_n$ such that the claim holds for every of the t_i. We conclude for $\delta_{p,P}$:

$$\begin{aligned}\delta'_{(p,P)}(t) &= \sum_{(r_1,P_1),\dots,(r_n,P_n)\in Q'} \delta'((p,P),f)((r_1,P_1),\dots,(r_n,P_n)) \prod_{i=1}^{n} \delta'_{(r_i,P_i)}(t_i) \\ &= \sum_{r_1,\dots,r_n\in Q} \delta(p,f)(r_1,\dots,r_n) \prod_{i=1}^{n} \sum_{P_i\subseteq Q\times\Sigma_0} d_\gamma(P_i)\, \delta'_{(r_i,P_i)}(t_i)\end{aligned}$$

By induction hypothesis we have $\sum_{P_i\subseteq Q\times\Sigma_0} d_\gamma(P_i)\, \delta'_{(r_i,P_i)}(t_i) = \delta_{r_i}(t_i)$ for all $i = 1,\dots,n$.

$$\begin{aligned}&= \sum_{r_1,\dots,r_n\in Q} \delta(p,f)(r_1,\dots,r_n) \prod_{i=1}^{n} \delta_{r_i}(t_i) \\ &= \delta_p(t).\end{aligned}$$

As $\delta'_{(p,P)}(t)$ does not depend on P at all, we obtain the claimed equality since d_γ is a distribution on $\mathcal{P}(Q \times \Sigma_0)$. Using this equation, we can now derive the behaviour of A':

$$\|A'\|(t) = \sum_{(p,P)\in Q'} \mu'(p,P)\, \delta'_{(p,P)}(t) = \sum_{p\in Q} \mu(p) \sum_{P\subseteq Q\times\Sigma_0} d_\gamma(P)\, \delta_{(p,P)}(t)$$

$$= \sum_{p \in Q} \mu(p)\, \delta_p(t) = \|A\|(t).$$

This completes the proof. ∎

We can shift the initial distribution to the final weights, thus obtaining an initial-normalized PTA with final weights. This is not possible with standard PTA.

Lemma 3.17. Let A be a PTA with final weights. There is a PTA with final weights A' such that the initial distribution μ' of A' is of the form $\mu' = \mathbb{1}_{\{q_\iota\}}$ for some state q_ι of A'.

Proof. Let $A = (Q, \delta, \mu, \gamma)$ and define $A' = (Q', \delta', \mu', \gamma')$ where

$$Q' = Q \cup \{q_\iota\}, \qquad \mu' = \mathbb{1}_{\{q_\iota\}}, \qquad \gamma'(q,a) = \begin{cases} \gamma(q,a) & \text{if } q \in Q \\ \|A\|(a) & \text{if } q = q_\iota, \end{cases}$$

$$\delta'(p,f)(q_1,\dots,q_n) = \begin{cases} \delta(p,f)(q_1,\dots,q_n) & \text{if } p, q_1, \dots, q_n \in Q \\ \sum_{q \in Q} \mu(q)\, \delta(q,f)(q_1,\dots,q_n) & \text{if } p = q_\iota \text{ and } q_1, \dots, q_n \in Q \\ 0 & \text{otherwise.} \end{cases}$$

Note that, by definition of A', we have $\delta'_q(t) = \delta_q(t)$ for all $q \in Q$ and $t \in \mathrm{T}_\Sigma$. Thus, we obtain the following behaviour of A' for a tree $t = f(t_1, \dots, t_n) \in \mathrm{T}_\Sigma$ with $n \geq 1$:

$$\begin{aligned} \|A'\|(t) = \delta_{q_\iota}(t) &= \sum_{q_1,\dots,q_n \in Q} \left(\sum_{q \in Q} \mu(q)\, \delta(q,f)(q_1,\dots,q_n) \right) \prod_{i=1}^{n} \delta'_{q_i}(t_i) \\ &= \sum_{q \in Q} \mu(q)\, \delta_q(t) = \|A\|(t). \end{aligned}$$

Finally, for a tree $t = a \in \Sigma_0$ we obtain $\|A'\|(a) = \gamma'(q_\iota, a) = \|A\|(a)$ directly by definition of A'. ∎

As in the word case, we show the correctness of a particular automata construction, that we will later use in the proof of Theorem 3.14.

Lemma 3.18. Let M be a finite set, d a distribution on M, $A = (Q, \delta, \mathbb{1}_{\{q_\iota\}}, \gamma)$ be a probabilistic tree automaton with final weights and $A' = (Q, T, q_\iota, F)$ a top-down deterministic and top-down complete tree automaton over $\Sigma \times M$ such that $\delta(p,f)(\overline{q}) = d(\{m \in M \mid (p,(f,m),\overline{q}) \in T\})$ for every $(p,f) \in Q \times \Sigma_n$ with $n \geq 1$, and $\gamma(p,a) = d(\{m \in M \mid (p,(a,m)) \in F\})$ for $(p,a) \in Q \times \Sigma_0$. Then,

$$\|A\|(t) = \mathrm{B}_p(\{u \in M^{\mathrm{pos}(t)} \mid (t,u) \in \mathrm{L}(A')\}),$$

for all $t \in \mathrm{T}_\Sigma$.

Proof. We show $\delta_q(t) = \mathrm{B}_p(\{u \in M^{\mathrm{pos}(t)} \mid (t,u) \in L_q\})$, where $L_q = \mathrm{L}(A'_q)$ and $A'_q = (Q, T, q, F)$ using induction on the tree height. Let $a \in \Sigma_0$. Then,

$$\begin{aligned}\delta_q(a) = \gamma(q,a) &= \sum_{m \in M} d(m)\, \mathbb{1}_F(q,(a,m)) = \sum_{m \in M} d(m)\, \mathbb{1}_L(a,m) \\ &= \mathrm{B}_p(\{u \in M^{\mathrm{pos}(t)} \mid (a,u) \in L_q\}).\end{aligned}$$

Next, consider the case $t = f(t_1, \ldots, t_n)$ with $n > 0$.

$$\delta_q(t) = \sum_{q_1,\ldots,q_n \in Q} \delta(q,f)(q_1,\ldots,q_n) \prod_{i=1}^{n} \delta_{q_i}(t_i)$$

We replace δ by its representation using T, use the induction hypothesis for $\delta_{q_i}(t_i)$:

$$= \sum_{q_1,\ldots,q_n \in Q} \sum_{m \in M} d(m)\, \mathbb{1}_T(p,(f,m),q_1,\ldots,q_n) \prod_{i=1}^{n} \sum_{u_i \in M^{\mathrm{pos}(t_i)}} d(u_i)\, \mathbb{1}_{L_{q_i}}(t_i,u_i)$$

Using distributivity, we merge the trees $u_i \in M^{\mathrm{pos}(t_i)}$ for $i = 1, \ldots, n$ and the symbol $m \in M$ into one tree $u \in M^{\mathrm{pos}(t)}$:

$$= \sum_{u \in M^{\mathrm{pos}(t)}} \mathrm{B}_d(\{u\}) \sum_{q_1,\ldots,q_n \in Q} \mathbb{1}_T(q,(f,u(\varepsilon)),q_1,\ldots,q_n) \prod_{i=1}^{n} \mathbb{1}_{L_{q_i}}(t_i, u|_i)$$

Since A' is deterministic and complete, the second sum collapses to $\mathbb{1}_L(t,u)$:

$$= \mathrm{B}_p(\{u \in M^{\mathrm{pos}(t)} \mid (t,u) \in L\}).$$

This completes the proof. ∎

We now have established all results we need for the proof of Theorem 3.14. The proof actually shows the equivalence of $\mathcal{N}_{\mathrm{D}}(\mathrm{T}_\Sigma)$ to PTA with final weights, which in turn are as expressive as probabilistic tree automata by Lemma 3.16.

Proof (of Theorem 3.14). Let S be the behaviour of a top-down probabilistic tree automaton. By Lemmas 3.16 and 3.17 there is a PTA with final weights $A = (Q, \delta, \mu, \gamma)$ such that $\|A\| = S$ and $\mu = \mathbb{1}_{q_\iota}$ for some state $q_\iota \in Q$. By Proposition 3.10 there is a finite set M, a distribution d on M, functions $\pi_{(p,f)} \colon M \to Q^n$ for every $(p,f) \in Q \times \Sigma^n$ and $n \geq 1$, and functions $\pi_{(p,a)} \colon M \to \{0,1\}$ for every $(p,a) \in Q \times \Sigma_0$ such that $\delta(p,f)(\bar{q}) = d(\pi^{-1}_{(p,f)}(\{\bar{q}\}))$ for all $(p,a) \in Q \times \Sigma^n$, $\bar{q} \in Q^n$, $n \geq 1$ and $\gamma(q,a) = d(\pi^{-1}_{(q,a)}(\{1\}))$ for all $(q,a) \in Q \times \Sigma_0$. For $(p,a) \in Q \times \Sigma_0$ we considered the distribution $d_{(p,a)}$ on $\{0,1\}$ with $d_{(p,a)}(1) = \gamma(p,a)$.

Let $\Gamma = \Sigma \times M$, and $g \colon \Gamma \to M$ and $h \colon \Gamma \to \Sigma$ be the canonical projections. We define the top-down deterministic and top-down complete tree automaton

$A' = (Q, T, q_\iota, F)$ by

$$T = \big\{(p, (f, m), \pi_{(p,f)(m)}) \;\big|\; p \in Q, (f, m) \in \Gamma\big\},$$
$$F = \big\{(q, (a, m)) \in Q \times \Gamma_0 \;\big|\; \pi_{(p,a)}(m) = 1\big\}.$$

Then, the automata A and A' satisfy the assumptions of Lemma 3.18. Therefore, $S = \|A\|(t) = (\mathrm{B}_d \circ g)(h^{-1}(\{t\}) \cap \mathrm{L}(A'))$ as claimed.

Conversely, assume $S \in \overleftarrow{\mathcal{N}}_{\mathrm{D}}(\mathrm{T}_\Sigma)$. Let Γ, M, d, g, h, L as in Definition 3.12. Let $\kappa \colon \Gamma \to \Sigma \times M$ be given by $\kappa(u) = (h(u), g(u))$. By definition of $\overleftarrow{\mathcal{N}}_{\mathrm{D}}(\mathrm{T}_\Sigma)$, we have that κ is injective. Thus, the tree language $\kappa(L) \subseteq \mathrm{T}_{\Sigma\times M}$ is also recognizable by a top-down deterministic and complete tree automaton A'. Let $A' = (Q, T, q_\iota, F)$. We construct a PTA with final weights over Σ by $A = (Q, \delta, \mathbb{1}_{\{q_\iota\}}, \gamma)$ with

$$\delta(p, f)(\bar{q}) = d(\{m \in M \mid (p, (f, m), \bar{q}) \in T\})$$
$$\gamma(q, a) = d(\{m \in M \mid (q, (a, m)) \in F\}),$$

for all $p \in Q$, $f \in \Sigma_n$, and $\bar{q} \in Q^n$. As before, the automata A and A' satisfy the requirements of Lemma 3.18 and we obtain $\|A\| = (\mathrm{B}_d \circ g)(h^{-1}(\{t\}) \cap \mathrm{L}(A')) = S$.∎

3.4.2 Nivat Classes and Bottom-Up Probabilistic Tree Automata

By Lemma 3.13 and Theorem 3.14 we know that top-down probabilistic tree automata are not powerful enough to describe every function in $\mathcal{N}(\mathrm{T}_\Sigma)$. In order to obtain a probabilistic automata model expressive equivalent to $\mathcal{N}(\mathrm{T}_\Sigma)$ we introduce bottom-up probabilistic tree automata. Whereas standard top-down probabilistic tree automata generalise top-down deterministic tree automata, bottom-up probabilistic tree automata generalise bottom-up deterministic tree automata. Though this step seems natural, the bottom-up model has gained very little interest before. In fact we could find just one other reference to it [L94].

Definition 3.19. A *bottom-up probabilistic tree automaton* is a triple $A = (Q, \delta, F)$ where

1. Q is a non-empty, finite set – the set of states,
2. $\delta = \bigcup_{n\geq 0} \delta_n$ where $\delta_n \colon \Sigma_n \times Q^n \to \Delta(Q)$ – the transition probabilities,
3. $F \subseteq Q$ – the set of final states.

For a tree $t \in \mathrm{T}_\Sigma$ we define the behaviour $\|A\|$ of A by

$$\|A\|(t) = \sum_{\substack{\rho\colon \mathrm{pos}(t)\to Q \\ \rho(\varepsilon)\in F}} \prod_{x\in\mathrm{pos}(t)} \delta(t(x), \rho(x_1), \ldots, \rho(xn_x))(\rho(x)),$$

where $n_x = \mathit{arity}(t(x))$.

As with top-down probabilistic-tree automaton, we can also define the behaviour inductively on the height of t: let distributions δ_t on Q be given by

$$\delta_t(q) = \sum_{q_1,\ldots,q_n\in Q} \delta(f, q_1, \ldots, q_n)(q) \prod_{i=1}^{n} \delta_{t_i}(q_i),$$

for all $t = f(t_1, \ldots, t_n) \in \mathrm{T}_\Sigma$ where $n \geq 0$. Note that the equation is also valid for $n = 0$. In this case the sum is just over the empty sequence of states, and $\delta_t(q)$ equals $\delta(t)(q)$ where $t \in \Sigma_0$. The behaviour $\|A\|$ of A on t is then given by

$$\|A\|(t) = \sum_{q\in F} \delta_t(q) = \delta_t(F).$$

Bottom-up probabilistic tree automata turn out to be exactly the right automata class to describe the tree series in $\mathcal{N}(\mathrm{T}_\Sigma)$.

Theorem 3.20. Let $S\colon \mathrm{T}_\Sigma \to [0,1]$ be a tree series. The following statements are equivalent.

1. $S = \|A\|$ for a bottom-up probabilistic tree automaton A.
2. $S \in \mathcal{N}(\mathrm{T}_\Sigma)$.

The translations are effective in both directions.

We show the behaviour of an automata construction that we use in the proof of Theorem 3.20.

Lemma 3.21. Let M be a finite set, d a distribution on M, $A = (Q, \delta, F)$ a bottom-up probabilistic tree automaton, and $A' = (Q, \delta', F)$ a bottom-up deterministic and bottom-up complete tree automaton such that $\delta(f, \overline{q})(p) = d(\{m \in M \mid \delta'((f, m), \overline{q}) = p\})$. Then,

$$\|A\|(t) = \mathrm{B}_d(\{u \in M^{\mathrm{pos}(t)} \mid (t, u) \in L\}),$$

for all $t \in \mathrm{T}_\Sigma$.

Proof. Let $t \in \mathrm{T}_\Sigma$ and $n_x = \mathit{arity}(t(x))$.

$$\|A\|(t) = \sum_{\substack{\rho \in Q^{\mathrm{pos}(t)} \\ \rho(\varepsilon) \in F}} \prod_{x \in \mathrm{pos}(t)} \delta(t(x), \rho(x_1), \ldots, \rho(x n_x))(\rho(x))$$

By the choice of M, d and δ' we obtain

$$= \sum_{\substack{\rho \in Q^{\mathrm{pos}(t)} \\ \rho(\varepsilon) \in F}} \prod_{x \in \mathrm{pos}(t)} \sum_{\substack{m \in M \\ \delta'((t(x),m),\rho(x1),\ldots,\rho(xn_x)) = \rho(x)}} d(m)$$

Rewriting the conditions on the indices as characteristic functions:

$$= \sum_{\rho \in Q^{\mathrm{pos}(t)}} \mathbb{1}_F(\rho(\varepsilon)) \prod_{x \in \mathrm{pos}(t)} \sum_{m \in M} d(m)\ \mathbb{1}_{\{\delta'((t(x),m),\rho(x1),\ldots,\rho(xn_x))\}}(\rho(x))$$

Using distributivity and commutativity we conclude

$$= \sum_{u \in M^{\mathrm{pos}(t)}} \left(\prod_{x \in \mathrm{pos}(t)} d(m) \right) \underbrace{\sum_{\rho \in Q^{\mathrm{pos}(t)}} \mathbb{1}_F(\rho(\varepsilon)) \prod_{x \in \mathrm{pos}(t)} \mathbb{1}_{\{\delta'((t(x),u(x)),\rho(x1),\ldots,\rho(xn_x))\}}(\rho(x))}_{= \mathbb{1}_L((t,u))}$$

Note that the second sum can only attain the values 0 or 1, since the automaton A' is deterministic and complete. We continue

$$\begin{aligned} &= \sum_{u \in M^{\mathrm{pos}(t)}} \mathrm{B}_d(\{u\})\, \mathbb{1}_L(t,u) \\ &= \mathrm{B}_d(\{u \in M^{\mathrm{pos}(t)} \mid (t,u) \in L\}) \\ &= (\mathrm{B}_d \circ g)(L \cap h^{-1}(\{t\})). \end{aligned}$$

∎

We are now ready to give the proof of Theorem 3.20.

Proof (of Theorem 3.20). The proof of both directions is similar to the proof of Theorem 3.14.

Let $S = \|A\|$ for a bottom-up probabilistic tree automaton $A = (Q, \delta, F)$. By Proposition 3.10 there is a finite, non-empty set M, a distribution d on M and functions $\pi_{(f,q_1,\ldots,q_n)} \colon M \to Q$ for all $f \in \Sigma_n$, $q_1, \ldots, q_n \in Q$, and $n \geq 0$ such that $\delta(f, q_1, \ldots, q_n)(q) = d(\pi^{-1}_{(f,q_1,\ldots,q_n)}(\{q\}))$ for all $q \in Q$. Define $\Gamma = \Sigma \times M$ and let $g \colon \Gamma \to M$ and $h \colon \Gamma \to \Sigma$ be the canonical projections. Furthermore, let the bottom-up deterministic and bottom-up complete tree automaton A' be given by $A' = (Q, \delta', F)$ where $\delta'((f,m), q_1, \ldots, q_n) = \pi_{(f,q_1,\ldots,q_n)}(m)$. Let $L = \mathrm{L}(A')$.

The automata A and A' satisfy the requirements of Lemma 3.21. Hence, we obtain $S(t) = \|A\|(t) = \mathrm{B}_p(\{u \in M^{\mathrm{pos}(t)} \mid (t,u) \in \mathrm{L}(A')\}) = (\mathrm{B}_p \circ g)(h^{-1}(\{t\}) \cap \mathrm{L}(A'))$.

Conversely, assume $S \in \mathcal{N}(\mathrm{T}_\Sigma)$. Let M, d, g, h, L as in Definition 3.12 such that (3.2) holds. Let $\kappa\colon \Gamma \to \Sigma \times M$ be given by $\kappa(u) = (h(u), g(u))$. As in the word case we obtain

$$S(t) = \mathrm{B}_p(\{u \in M^{\mathrm{pos}(t)} \mid (t,u) \in \kappa(L)\}).$$

Let $A' = (Q, \delta', F)$ be a bottom-up deterministic and bottom-up complete tree automaton with $\mathrm{L}(A') = \kappa(L)$. Note that this automaton always exists as bottom-up tree automata are determinisable. We define a bottom-up probabilistic tree automaton A by $A = (Q, \delta, F)$ and $\delta(f, \overline{q})(q) = d(\{m \in M \mid \delta'((f,m), \overline{q}) = q\})$ for all $f \in \Sigma_n$ and $\overline{q} \in Q^n$. Again, by Lemma 3.21, we obtain $\|A\|(t) = \mathrm{B}_p(\{u \in M^{\mathrm{pos}(t)} \mid (t,u) \in \mathrm{L}(A')\}) = \mathrm{B}_p(\{u \in M^{\mathrm{pos}(t)} \mid (t,u) \in \kappa(L)\}) = S(t)$. This completes the proof. ∎

Corollary 3.22. The class of tree series recognizable by top-down probabilistic tree automata is contained in the class of tree series recognizable by bottom-up probabilistic tree automata.

Furthermore, if Σ contains at least one symbol with arity at least 2 and at least one symbol with arity 0, the inclusion is strict.

Proof. Let A_T be a top-down probabilistic tree automaton, by Theorem 3.14, we have $\|A_\mathrm{T}\| \in \mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma)$. As $\mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma) \subseteq \mathcal{N}(\mathrm{T}_\Sigma)$, we obtain the existence of a bottom-up probabilistic tree automaton A_B with $\|A_\mathrm{B}\| = \|A_\mathrm{T}\|$ by Theorem 3.20.

Now assume there is a symbol $f \in \Sigma$ with $\mathit{arity}(f) \geq 2$. By Lemma 3.13 we know there is a tree series $S \in \mathcal{N}(\mathrm{T}_\Sigma) \setminus \mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma)$. Again by Theorem 3.20 we conclude there is a bottom-up PTA A_B with $\|A_\mathrm{B}\| = S$. Assume there is also a top-down PTA A_T with $\|A_\mathrm{T}\| = S$. By Theorem 3.14 this implies $S \in \mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma)$. A contradiction. ∎

Chapter 4

Classical MSO Logic

Predicate logic can be considered as the lingua franca of mathematics and also of theoretical computer science. An important fragment of predicate logic is monadic second order (MSO) logic, where quantification is allowed over elements of the domain as well as over subsets of the domain, but not over relations of arity greater than two.

In this chapter, we will recall the classical definition of MSO logic over arbitrary signatures and give the corresponding semantics. We will also show how to apply these definitions to words and trees. At the end of the chapter, we are ready to recall Büchi's famous theorem stating the equivalence of MSO definable languages and recognizable languages.

4.1 Signatures and Structures

Before we introduce MSO logic itself we define signatures and structures, which will later be used to give a general definition of MSO logic and probabilistic MSO logic independent of the actual domain and relations. For an in depth introduction to model theory see for example [CK12].

Definition 4.1. A *signature* $\mathcal{S} = (S, arity)$ consists of [1]

1. A set of relation symbols S,
2. A function $arity\colon S \to \mathbb{N}$ assigning an arity to every relation symbol.

Definition 4.2. Let $\mathcal{S} = (S, arity)$ be a signature. A *$\mathcal{S}$-structure* is a tuple $\mathcal{A} = (A, (R^{\mathcal{A}})_{R \in S})$ where

1. A is a set – the carrier set,
2. $R^{\mathcal{A}} \subseteq A^{arity(R)}$ is an $arity(R)$-ary relation over A for every $R \in S$.

[1] To avoid confusion with "σ-algebra", we use $\mathcal{S}$ for signature instead of σ.

If we are only interested in the carrier set, or domain, of A, we write $\mathrm{dom}(\mathcal{A})$ for A.

As we are interested in the MSO formulas that work on words and trees, we give the signatures and structures used to describe a single word or a single tree below.

The Structure of Words

Let Σ be a finite alphabet. The word signature $\mathcal{W}_\Sigma$ is given by $\mathcal{W}_\Sigma = (\{\leq\} \cup \{\mathrm{label}_a \mid a \in \Sigma\}, \mathit{arity})$ with $\mathit{arity}(\leq) = 2$ *and* $\mathit{arity}(\mathrm{label}_a) = 1$ for all $a \in \Sigma$. For a finite or infinite word $w = (w_x)_{x \in \mathrm{pos}(w)} \in \Sigma^\infty$, we define a $\mathcal{W}_\Sigma$-structure $\widetilde{w}$ by

$$\widetilde{w} = \big(\mathrm{pos}(w), \leq|_{\mathrm{pos}(w)^2}, (\mathrm{label}_a^w)_{a \in \Sigma}\big),$$

where $\leq$ is the usual order on the integers, and $\mathrm{label}_a^w = \{x \in \mathrm{pos}(w) \mid w_x = a\}$. It is easy to see, that $\widetilde{w}$ describes the word w uniquely.

The Structure of Trees

Now, assume Σ is a finite ranked alphabet. Let $N = \max\{n \geq 0 \mid \Sigma_n \neq \emptyset\}$. The tree signature $\mathcal{T}_\Sigma$ is given by

$$\mathcal{T}_\Sigma = \big(\{\mathrm{edge}_i, \mathrm{label}_a \mid i = 1, \ldots, N,\ a \in \Sigma\}, \mathit{arity}\big),$$

where $\mathit{arity}(\mathrm{edge}_i) = 2$ and $\mathit{arity}(\mathrm{label}_a) = 1$ for every $1 \leq i \leq N$ and $a \in \Sigma$. The unary relations label_a serve the same purpose as in the word case, whereas the relations edge_i model the branching structure of the tree. Formally, given a finite tree $t \in \mathrm{T}_\Sigma$, we define the $\mathcal{T}_\Sigma$-structure $\widetilde{t}$ by

$$\widetilde{t} = \big(\mathrm{pos}(t), (\mathrm{edge}_i^t)_{i=1,\ldots,N}, (\mathrm{label}_a^t)_{a \in \Sigma}\big),$$

where

$$\mathrm{edge}_i^t = \big\{(x, xi) \in \mathrm{pos}(t)^2 \;\big|\; x \in \mathrm{pos}(t)\, 1 \leq i \leq N\big\},$$
$$\mathrm{label}_a^t = \big\{x \in \mathrm{pos}(t) \;\big|\; t(x) = a\big\}.$$

4.2 Syntax and Semantics of MSO Logic

For the rest of this chapter fix two countable, disjoint sets $\mathcal{V}_1$ and $\mathcal{V}_2$ and let $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$. These sets contain the symbols which will be used as first order, second-order, respectively, variable symbols in MSO logic. For the definition of the semantics of an MSO formula, we need to assign values to these symbols: given an $\mathcal{S}$-structure $\mathcal{A} = (A, (R^\mathcal{A})_{R \in S})$, we say that a function $\alpha \colon \mathcal{V} \to A \cup \mathcal{P}(A)$ is an *$\mathcal{A}$-assignment* if $\alpha(\mathcal{V}_1) \subseteq A$ and $\alpha(\mathcal{V}_2) \subseteq \mathcal{P}(A)$. For any $\mathcal{A}$-assignment α, variable $x \in \mathcal{V}_1$, and value

$a \in A$, we denote by $\alpha[x \mapsto a]$ the updated assignment α', which assigns x to a and agrees with α everywhere else. Likewise, $\alpha[X \mapsto M]$ denotes the update for a second order variable $X \in \mathcal{V}_2$ by the subset $M \subseteq A$.

Definition 4.3. Let $\mathcal{S} = (S, \mathit{arity})$ be a signature. The set of all MSO formulas φ over $\mathcal{S}$ is given in BNF by

$$\varphi ::= R(x_1, \ldots, x_{\mathit{arity}(s)}) \mid x_1 = x_2 \mid x \in X \mid \varphi \wedge \varphi \mid \neg\varphi \mid \forall x.\varphi \mid \forall X.\varphi\,,$$

where $R \in S$, $x, x_1, x_2, \ldots \in \mathcal{V}_1$ and $X \in \mathcal{V}_2$. The set of all MSO formulas over $\mathcal{S}$ is denoted by $\mathrm{MSO}(\mathcal{S})$.

Given a $\mathcal{S}$-structure $\mathcal{A}$ and an $\mathcal{A}$-assignment α, we define the satisfaction relation $(\mathcal{A}, \alpha) \models \varphi$ inductively on the structure of φ:

$$\begin{aligned}
(\mathcal{A}, \alpha) &\models R(x_1, \ldots, x_{\mathit{arity}(s)}) && \iff (\alpha(x_1), \ldots, \alpha(x_{\mathit{arity}(s)})) \in R^{\mathcal{A}},\\
(\mathcal{A}, \alpha) &\models x_1 = x_2 && \iff \alpha(x_1) = \alpha(x_2),\\
(\mathcal{A}, \alpha) &\models x \in X && \iff \alpha(x) \in \alpha(X),\\
(\mathcal{A}, \alpha) &\models \varphi_1 \wedge \varphi_2 && \iff (\mathcal{A}, \alpha) \models \varphi_1 \text{ and } (\mathcal{A}, \alpha) \models \varphi_2,\\
(\mathcal{A}, \alpha) &\models \neg\varphi && \iff (\mathcal{A}, \alpha) \not\models \varphi,\\
(\mathcal{A}, \alpha) &\models \forall x.\varphi && \iff (\mathcal{A}, \alpha[x \mapsto m]) \models \varphi \text{ for all } m \in A,\\
(\mathcal{A}, \alpha) &\models \forall X.\varphi && \iff (\mathcal{A}, \alpha[X \mapsto M]) \models \varphi \text{ for all } M \subseteq A.
\end{aligned}$$

We associate with every MSO formula φ the set of *free variables* used in φ. The inductive definition is as follows:

$$\begin{aligned}
\mathrm{Free}(x_1 = x_2) &= \{x_1, x_2\}, & \mathrm{Free}(s(x_1, \ldots, x_{\mathit{arity}(s)})) &= \{x_1, \ldots, x_{\mathit{arity}(s)}\},\\
\mathrm{Free}(x \in X) &= \{x, X\}, & \mathrm{Free}(\varphi_1 \wedge \varphi_2) &= \mathrm{Free}(\varphi_1) \cup \mathrm{Free}(\varphi_2),\\
\mathrm{Free}(\neg\varphi) &= \mathrm{Free}(\varphi), & \mathrm{Free}(\forall x.\varphi) &= \mathrm{Free}(\varphi) \setminus \{x\},\\
\mathrm{Free}(\forall X.\varphi) &= \mathrm{Free}(\varphi) \setminus \{X\}.
\end{aligned}$$

It can be shown that $(\mathcal{A}, \alpha) \models \varphi \iff (\mathcal{A}, \tau) \models \varphi$ holds if $\alpha|_{\mathrm{Free}(\varphi)} = \tau|_{\mathrm{Free}(\varphi)}$. We call a MSO formula φ a *sentence* if $\mathrm{Free}(\varphi) = \emptyset$. Thus, satisfaction of a MSO sentence does not depend on the assignment at all. Hence, for a MSO sentence φ, we just write $\mathcal{A} \models \varphi$ if $(\mathcal{A}, \alpha) \models \varphi$ for any $\mathcal{A}$-assignment α.

In order to define the language defined by a $\mathrm{MSO}(\mathcal{S})$ sentence φ, we fix a set of $\mathcal{S}$-structures C and define the language of φ relative in this set. Formally, for a set C of $\mathcal{S}$-structures and a MSO sentence φ, let the *language defined by* φ be

$$\mathrm{L}_C(\varphi) = \{\mathcal{A} \in C \mid \mathcal{A} \models \varphi\}.$$

If C is understood, we just write $\mathrm{L}(\varphi)$. We say a language $L \subseteq C$ is *definable* if there is a MSO sentence φ with $L = \mathrm{L}_C(\varphi)$.

When considering the structures that arise from words or finite trees as introduced in Section 4.1, we identify the $\mathcal{W}_\Sigma$-structure $\widetilde{w}$ with the word w itself. Thus, for a MSO($\mathcal{W}_\Sigma$)-sentence φ, we regard $\mathrm{L}_C(\varphi)$, where $C = \{\widetilde{w} \mid w \in \Sigma^\infty\}$, as a subset of Σ^∞. Likewise, we identify the $\mathcal{T}_\Sigma$-structure $\widetilde{t}$ with the tree t itself, and consider $\mathrm{L}_{C'}(\varphi')$, where $C' = \{\widetilde{t} \mid t \in \mathrm{T}_\Sigma\}$, as set of finite trees.

Example 4.4. We want to describe the language L from Examples 2.2 and 6.4 using an MSO formula φ. Recall that $L = \mathrm{a}(\mathrm{bb} \cup \mathrm{a})^* \cup \mathrm{a}(\mathrm{a}^*\mathrm{b})^\omega$. We split the formula in two parts, one for finite words, one for infinite words. The formula is not as succinct as the regular expression. We noted below parts of the formula their intuitive semantics. We define two separate formulas: one for the finite word part and one for the infinite word part.

$$\begin{aligned}
\varphi_1 = {} & \underbrace{(\exists y.\forall x.x \le y)}_{\text{finite word}} \wedge \underbrace{(\exists x.\, \mathrm{label}_\mathrm{a}(x) \wedge \forall y.x \le y)}_{\text{first letter is a}} \\
& \wedge \forall X.\Big(\underbrace{\big(cib(X) \wedge \forall Y.cib(Y) \implies (\forall x.x \in Y \implies x \in X)\big)}_{X \text{ is maximal consequence sequence of b's}} \\
& \qquad \implies \exists Y.\Big(\underbrace{\big(\exists x.x \in X \wedge x \in Y \wedge \forall y.y \in X \implies x \le y\big)}_{\text{the first position of } X\ (=u) \text{ is in } Y} \\
& \qquad\quad \wedge \underbrace{\big(\forall x.\forall y.y = x+1 \implies (x \in Y \iff y \notin Y)\big)}_{\text{exactly every second position, counted from } u, \text{ is in } Y} \\
& \qquad\quad \wedge \underbrace{\big(\exists x.x \in X \wedge x \notin Y \wedge (\forall y.y \in X \implies y \le x)\big)}_{\text{last position in } X \text{ is not in } Y}\Big)\Big) \\
\varphi_2 = {} & \underbrace{\forall x.\exists y.y \ge x \wedge y \ne x \wedge \mathrm{label}_\mathrm{b}(y)}_{\text{infinitely many b labelled positions}} \wedge \underbrace{\exists x.\, \mathrm{label}_\mathrm{a}(x) \wedge \forall y.x \le y}_{\text{first position is labelled with a}}
\end{aligned}$$

We used the following abbreviations for formulas (*cib* – closed interval of b's):

$$\begin{aligned}
cib(X) = {} & \big(\exists x.\exists y.\big((\forall z.(x \le z \wedge z \le y) \iff z \in X)\big) \\
& \qquad\qquad \wedge (\forall z.z \in X \implies \mathrm{label}_\mathrm{b}(z))\big), \\
\big(y = x+1\big) = {} & \big(x \ne y \wedge x \le y \wedge \forall z(x \le z \wedge z \le y) \implies (z = x \vee z = y)\big).
\end{aligned}$$

Note that any set M which contains every second position starting from some position x, contains exactly the positions with even distance from x. Thus, the word from position x to some position $y \in M$ including y, has odd length. Defining $\varphi = \varphi_1 \wedge \varphi_2$ yields the desired formula with $\mathrm{L}(\varphi) = L$.

Example 4.5. As in the previous example, we want to give an MSO sentence for the tree language from Examples 2.22 and 7.3, i.e. the language L of all languages over $\Sigma = \{\mathrm{f}, \mathrm{a}, \mathrm{b}\}$ with at least one a labelled node. Such an MSO sentence φ can easily be given:

$$\varphi = \exists x.\, \mathrm{label}_{\mathrm{a}}(x).$$

Note that, in contrast to the previous example, the MSO formula is much shorter than the regular tree expression.

A famous result by J. R. Büchi states that the languages definable by MSO sentences over words are exactly the recognizable word languages. The original statement became known as Büchi's theorem [B60]. Other versions are provided by Elgot [E61] and Trakhtenbrot [T61].

Theorem 4.6 (Büchi's theorem). Let $L \subseteq \Sigma^*$. The following statements are equivalent

1. L is recognizable.
2. $L = \mathrm{L}(\varphi)$ for some $\mathrm{MSO}(\mathcal{W}_\Sigma)$-sentence φ.

The same statement also holds in the setting of finite trees. This has been shown by Thatcher and Wright [TW68].

Theorem 4.7. Let $L \subseteq \mathrm{T}_\Sigma$. The following statements are equivalent:

1. L is recognizable.
2. $L = \mathrm{L}(\varphi)$ for some $\mathrm{MSO}(\mathcal{T}_\Sigma)$-sentence φ.

Chapter 5

Probabilistic MSO Logic

In this chapter we extend MSO logic from Chapter 4 to a probabilistic logic. We do so by adding probability constants and a new "expected value" second order quantifier to the logic.

In Section 5.1, we introduce a σ-algebra on sets of positions and transfer Bernoulli measures, that were introduced in Section 3.2, to this algebra. With these definitions set up, we can define the syntax and semantics of probabilistic MSO logic in Section 5.2. This syntax is extended in Section 5.3 by additional first order quantifiers which do not add expressive power to the logic but allows us to write certain formulas more succinctly. Finally, we show the equivalence of probabilistic MSO logic and Nivat-classes in Section 5.4.

The results on words have been published in [W12] and the results on finite trees in [W15].

5.1 Measuring Sets of Positions

In Section 3.2 we defined Bernoulli measures on words and trees over finite (ranked) alphabets. As the objects in MSO logic are not words, but subsets of an arbitrary domain, we give a definition of Borel-σ-algebra and Bernoulli measure that works on sets. For countable structures, we can assume an enumeration of the structure and define a metric similar to the metric on infinite words, c.f. Definition 2.6.

Let A be a countable set and fix an enumeration $E = (a_1, a_2, \dots)$ of A. We define a metric d_E on $\mathcal{P}(A)$ by

$$d_E(X, Y) = \begin{cases} 2^{-\min\{\, i \geq 1 \mid a_i \in X \triangle Y \,\}} & \text{if } X \neq Y \\ 0 & \text{if } X = Y, \end{cases}$$

where $X \triangle Y$ denotes the symmetric difference of X and Y. With this definition $(\mathcal{P}(A), d_E)$ becomes a compact metric space. Thus, we can apply Definition 2.7, and define the Borel-σ-algebra $\mathcal{B}(\mathcal{P}(A), d_E)$ over $\mathcal{P}(A)$. We will later see that this σ-algebra does not depend on the enumeration E.

Similar to Lemma 2.8 one shows that $\mathcal{B}(\mathcal{P}(A), d_E)$ is generated by the cylinder sets of the following form:

$$\mathrm{Cyl}_E^n(X) = \big\{ Y \subseteq A \;\big|\; Y \cap \{a_1, \ldots, a_n\} = X \big\},$$

for $X \subseteq \{a_1, \ldots, a_n\}$. The system of all cylinder sets is intersection closed. Thus, two probability measures that agree on all cylinder sets are already equal.

Using the cylinder sets, we can transfer the notion of Bernoulli measure, as introduced in Section 3.2, to the subsets of A. Let $p \in [0, 1]$, we define the measure $\mathrm{B}_{p,E}^{\mathcal{P}(A)}$ on $\mathcal{B}(\mathcal{P}(A), d_E)$ by

$$\mathrm{B}_{p,E}^{\mathcal{P}(A)}(\mathrm{Cyl}_E^n(X)) = p^{|X|}\,(1-p)^{n-|X|}.$$

The existence and uniqueness of a measure $\mathrm{B}_{p,E}^{\mathcal{P}(A)}$ follows from standard measure theory: either write $\mathrm{B}_{p,E}^{\mathcal{P}(A)}$ as countable product measure of a binary distribution, or apply Carathéodory's extension theorem directly, see [K08] for details.

As usual, if A is understood from the context, we just write $\mathrm{B}_{p,E}$ for $\mathrm{B}_{p,E}^{\mathcal{P}(A)}$.

Up to now, the σ-algebra as well as the measure $\mathrm{B}_{p,E}$ depend on the choice of the enumeration E. Whereas the metric d_E certainly depends on E, we show that $\mathcal{B}(\mathcal{P}(A), d_E)$ and $\mathrm{B}_{p,E}$ actually do not.

Lemma 5.1. Let A be a countable set and E, E' be two enumerations of A. Then $\mathcal{B}(\mathcal{P}(A), d_E) = \mathcal{B}(\mathcal{P}(A), d_{E'})$ and $\mathrm{B}_{p,E} = \mathrm{B}_{p,E'}$.

Proof. Let $E = (a_1, a_2, \ldots)$ and $E' = (a'_1, a'_2, \ldots)$. We show that every cylinder set $\mathrm{Cyl}_E^n(X)$ is contained in $\mathcal{B}(\mathcal{P}(A), d_{E'})$. Let $N > 0$ such that $\{a_1, \ldots, a_n\} \subseteq \{a'_1, \ldots, a'_N\}$. We have

$$\mathrm{Cyl}_E^n(X) = \bigcup_{\substack{X' \subseteq \{a'_1, \ldots, a'_N\} \\ X' \cap \{a_1, \ldots, a_n\} = X}} \mathrm{Cyl}_{E'}^N(X'). \tag{5.1}$$

Thus, $\mathrm{Cyl}_E^n(X) \in \mathcal{B}(\mathcal{P}(A), d_{E'})$ for every $n \geq 1$ and $X \subseteq A$. Hence, $\mathcal{B}(\mathcal{P}(A), d_E) \subseteq \mathcal{B}(\mathcal{P}(A), d_{E'})$, as the cylinder sets generate $\mathcal{B}(\mathcal{P}(A), d_E)$. By exchanging primed and unprimed symbols, one proves $\mathcal{B}(\mathcal{P}(A), d_{E'}) \subseteq \mathcal{B}(\mathcal{P}(A), d_E)$. This shows the first part of the lemma.

To show $\mathrm{B}_{p,E} = \mathrm{B}_{p,E'}$, we prove that the equality holds on all cylinder sets $\mathrm{Cyl}_E^n(X)$. We use the representation from (5.1). Note that the union in (5.1) is over pairwise

disjoint sets.

$$\begin{aligned}\mathrm{B}_{p,E'}\big(\mathrm{Cyl}_E^n(X)\big) &= \sum_{\substack{X'\subseteq\{a'_1,\dots,a'_N\}\\ X'\cap\{a_1,\dots,a_n\}=X}} \mathrm{B}_{p,E'}(\mathrm{Cyl}_{E'}^N(X')) \\ &= \sum_{\substack{X'\subseteq\{a'_1,\dots,a'_N\}\\ X'\cap\{a_1,\dots,a_n\}=X}} p^{|X'|}\,(1-p)^{N-|X'|}\end{aligned}$$

Every set X' must satisfy $X \subseteq X'$ and $\{a_1,\dots,a_n\} \setminus X \subseteq \{a_1,\dots,a_N\} \setminus X'$. Thus, $|X|$ elements are fixed in X' and $n-|X|$ entries are fixed in X^{C}. We continue

$$\begin{aligned}&= p^{|X|}\,(1-p)^{n-|X|} \\ &= \mathrm{B}_{p,E}\big(\mathrm{Cyl}_E^n(X)\big).\end{aligned}$$

As the system of cylinder sets $\mathrm{Cyl}_E^n(X)$ is an intersection-closed generating system of $\mathcal{B}(\mathcal{P}(A), d_E)$, we obtain $\mathrm{B}_{p,E} = \mathrm{B}_{p,E'}$. ∎

By Lemma 5.1, we can omit the index "E" and just write $\mathcal{B}(\mathcal{P}(A))$ and B_p where we assume an arbitrary enumeration on A.

The Bernoulli measures on powersets introduced here and the Bernoulli measures on words and trees introduced in Section 3.2 are connected via the characteristic function and the support function, respectively.

Lemma 5.2. For any set D, let $c\colon \mathcal{P}(D) \to \{0,1\}^D$ map any subset to its characteristic function. The following statements hold:

1. Let $D = \{1,\dots,n\}$ for some $n \in \mathbb{N}$, or $D = \mathbb{N}$ and $n = \omega$. Then, it holds that $c^{-1}(\mathcal{B}(\{0,1\}^N)) = \mathcal{B}(\mathcal{P}(D))$ and $\mathrm{B}_p^n = \mathrm{B}_p^{\mathcal{P}(D)} \circ c^{-1}$.

2. Let D be a finite tree domain. Then, it holds that $c^{-1}(\mathcal{B}(\{0,1\}^N)) = \mathcal{B}(\mathcal{P}(D))$ and $\mathrm{B}_p^D = \mathrm{B}_p^{\mathcal{P}(D)} \circ c^{-1}$.

The measures B_p^n and B_p^D denote the ones introduced in Section 3.2.

Proof. We first consider the finite cases. Since the Borel-σ-algebra is just the whole powerset of D and c is bijective, we immediately obtain that the Borel-σ-algebras transfer. The proof for the statement 2. is analogous to the case $D = \mathbb{N}$, which is given below.

Assume $D = \mathbb{N}$ and $E = (1,2,\dots)$ the canonical enumeration of D. Let $u_1 \cdots u_k \in \{0,1\}^*$. By definition, we have $c^{-1}(u_1 \cdots u_k\{0,1\}^\omega) = \mathrm{Cyl}_E^k(X)$ where $X = \{i \in \{1,\dots,k\} \mid u_i = 1\}$. Thus, $\mathcal{B}(\mathcal{P}(\mathbb{N})) \subseteq c^{-1}(\mathcal{B}(\{0,1\}^\omega))$. Conversely, for every

cylinder set $\mathrm{Cyl}_E^k(X)$ for some set $X \subseteq \{1, \dots, k\}$ define $u_i = \mathbb{1}_X(i)$ for $i = 1, \dots, k$. Then, $\mathrm{Cyl}_E^k(X) = c^{-1}(u_1 \cdots u_k\{0,1\}^\omega)$ and thus $\mathcal{B}(\mathcal{P}(\mathbb{N})) = c^{-1}(\mathcal{B}(\{0,1\}^\omega))$.

We show $\mathrm{B}_p^n = \mathrm{B}_{p,E}^{\mathcal{P}(\mathbb{N})} \circ c^{-1}$. Let $u_1 \cdots u_k\{0,1\}^\omega$ be a cylinder set in $\mathcal{B}(\{0,1\}^\omega)$. Let $d_p \colon \{0,1\} \to [0,1]$ with $d_p(1) = p$ and $d_p(0) = 1 - p$. We conclude

$$\begin{aligned}
\mathrm{B}_p^\omega(u_1 \cdots u_k\{0,1\}^\omega) &= \prod_{i=1}^{k} d_p(u_i) \\
&= p^{|X|}(1-p)^{k-|X|} \quad \text{where } X = \{i \in \{1, \dots, k\} \mid u_i = 1\} \\
&= \mathrm{B}_{p,E}^{\mathcal{P}(\mathbb{N})}(\mathrm{Cyl}_E^k(X)) \\
&= \mathrm{B}_{p,E}^{\mathcal{P}(\mathbb{N})}(c^{-1}(u_1 \cdots u_k\{0,1\}^\omega)).
\end{aligned}$$

This completes the proof. ∎

The definitions above allow us to handle countable domains only. While this is a restriction, most interesting structures in computer science have a countable domain: all finite structure, infinite words, or infinite trees. Therefore, we assume for the rest of this chapter that every considered structure is countable. Since, our probabilistic logic will permit application of a probability measure to a definable set of subsets, we make the following assumption to ensure well-definedness.

Assumption 5.3. Let $\mathcal{A}$ be a $\mathcal{S}$-structure with countable carrier set A. We say that *definable sets are measurable in* $\mathcal{A}$ if for every $n \geq 1$, MSO formula φ and $\mathcal{A}$-assignment α the set

$$\left\{(M_1, \dots, M_n) \in (2^A)^n \;\middle|\; (\mathcal{A}, \alpha[X_1 \mapsto M_1, \dots, X_n \mapsto M_n]) \models \varphi\right\}$$

is measurable in $\bigotimes_{i=1}^n \mathcal{B}(\mathcal{P}(A))$ for all $X_1, \dots, X_n \in \mathcal{V}_2$ and $n \geq 1$.

From now on, for the rest of this chapter, we only consider countable structures, where every tuple of definable sets is also measurable. Again:

We assume that every structure is countable and, that definable sets are also measurable.

It can be shown that every such set is a so called projective set, i.e., built from a Borel set in some Polish space using projection and complement. Those sets are universally measurable, if the axiom of projective determinacy (PD) is assumed. Fortunately, we only consider cases, where we can directly show that every definable set is measurable without additional axioms.

Proposition 5.4. In the case of finite or infinite words and finite trees, every definable set is measurable.

Proof. The statement is trivial in the finite case, as $\mathcal{B}(\mathcal{P}(A))$ is just $\mathcal{P}(A)$ in this case and every subset of A is measurable. For infinite words, let φ be an MSO formula, and $X_1, \dots, X_n \in \mathcal{V}_2$. Let $\mathcal{V}' = \mathrm{Free}(\varphi) \cup \{X_1, \dots, X_n\}$ and $\mathcal{V}'' = \mathrm{Free}(\varphi) \setminus \{X_1, \dots, X_n\}$, i.e., $\mathcal{V}' = \mathcal{V}'' \cup \{X_1, \dots, X_n\}$ and the sets are disjoint. We encode a pair (w, α), where α is a w-assignment, as word over $N_{\mathcal{V}'} = \Sigma \times \{0,1\}^{\mathcal{V}'}$ as usual: the additional components in $N_{\mathcal{V}'}$ mark the positions which are included in the subsets, the position which is assigned to a first order variable, respectively. By (the proof of) Büchi's theorem the language $L \subseteq N_{\mathcal{V}'}^{\omega}$, which contains all encoded pairs (w, α) with $(w, \alpha) \models \varphi$, is regular.

We apply Corollary 3.8 with $\Sigma' = \Sigma \times \{0,1\}^{\mathcal{V}''}$, $M = \{0,1\}^{\{X_1,\dots,X_n\}}$, $\Gamma = \Sigma' \times M = N_{\mathcal{V}'}$, and $g \colon \Gamma \to M$ and $h \colon \Gamma \to \Sigma$ the canonical projections. This yields that the set

$$\left\{(u_1, \dots, u_n) \in (\{0,1\}^n)^{\omega} \;\middle|\; (w, \alpha[X_1 \mapsto \mathrm{supp}(u_1), \dots, X_n \mapsto \mathrm{supp}(u_n)]) \models \varphi\right\}$$

is measurable, where supp maps every word $(u_i)_{i \geq 1} \in \{0,1\}^{\omega}$ to the set of positions i with $u_i = 1$. Since $(\{0,1\}^n)^{\omega}$ and $(\{0,1\}^{\omega})^n$ are homeomorphic to each other, i.e., there is a continuous bijective function which has a continuous inverse function, and the characteristic function $c \colon \mathcal{P}(\mathbb{N}) \to \{0,1\}^{\omega}$ is Borel-measurable, we conclude that $\{(M_1, \dots, M_n) \in (\mathcal{P}(\mathbb{N}))^n \mid (w, \alpha[X_i \mapsto M_i]_{i=1}^n \models \varphi\}$ is measurable. ∎

Assumption 5.3 even holds for infinite trees. This result has recently been shown by Gogacz, Michalewski, Mio and Skrzypczak [GMMS14].

5.2 Syntax and Semantics of Probabilistic MSO Logic

At the beginning of this section, we give the definition of the syntax and the semantics of probabilistic MSO logic. Afterwards, we give some basic semantic equivalences and derive a normal form for probabilistic MSO formulas.

Definition 5.5. Let $\mathcal{S} = (S, \mathit{arity})$ be a signature. The set $\mathrm{PMSO}(\mathcal{S})$ of all *probabilistic MSO formulas* φ over $\mathcal{S}$ is given in BNF by

$$\varphi ::= \psi \mid p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbb{E}_p X.\varphi,$$

where ψ is an $\mathrm{MSO}(\mathcal{S})$ formula, p a probability value, and X a second order variable.

Let C be a set of $\mathcal{S}$-structures. We define the semantics of a probabilistic MSO formula in φ in C as a function $[\![\varphi]\!]_C$ mapping a $\mathcal{S}$-structure $\mathcal{A} \in C$ with $\mathcal{A} = (A, (s^{\mathcal{A}})_{s\in S})$ and an $\mathcal{A}$-assignment α to a probability value. If C is understood, we just write $[\![\varphi]\!]$ for $[\![\varphi]\!]_C$. Formally we define

$$
\begin{aligned}
[\![\psi]\!](\mathcal{A}, \alpha) &= \begin{cases} 1 & \text{if } (\mathcal{A}, \alpha) \models \psi \\ 0 & \text{otherwise,} \end{cases} \\
[\![p]\!](\mathcal{A}, \alpha) &= p \\
[\![\varphi_1 \wedge \varphi_2]\!](\mathcal{A}, \alpha) &= [\![\varphi_1]\!](\mathcal{A}, \alpha) \cdot [\![\varphi_2]\!](\mathcal{A}, \alpha), \\
[\![\neg\varphi]\!](\mathcal{A}, \alpha) &= 1 - [\![\varphi]\!](\mathcal{A}, \alpha), \\
[\![\mathbb{E}_p\, X.\varphi]\!](\mathcal{A}, \alpha) &= \int_{M\subseteq A} [\![\varphi]\!](\mathcal{A}, \alpha[X \mapsto M])\ \mathrm{B}_p^A(\mathrm{d}M).
\end{aligned}
$$

In the case of finite structures, no measure theory is necessary to define the semantics of $\mathbb{E}_p\, X.\varphi$: assume A is finite, then

$$
[\![\mathbb{E}_p\, X.\varphi]\!](\mathcal{A}, \alpha) = \sum_{M\subseteq A} [\![\varphi]\!](\mathcal{A}, \alpha[X \mapsto M]) \cdot p^{|M|}(1-p)^{|A\setminus M|}.
$$

The semantics of conjunction and negation are motivated from probability theory as these correspond to the probability of the intersection of independent events, respectively, to the probability of the complement of an event.

We still need to show that the semantics given in Definition 5.5 is well-defined, i.e., the integral in the semantics of $\mathbb{E}_p\, X.\varphi$ is only applied to measurable functions and attains only values in $[0, 1]$. The second statement is an easy consequence of this first one: if $[\![\varphi]\!]$ is bounded by 1, one obtains, by monotonicity of the integral, $\|\mathbb{E}_p\, X.\varphi\| \leq \int 1\ \mathrm{dB}_p^A = 1$. We show the measurability claim.

Lemma 5.6. Let φ be a probabilistic MSO formula, $\mathcal{A}$ be an $\mathcal{S}$-structure, α an $\mathcal{A}$-assignment, and $X_1, \ldots, X_n$ second order variable symbols. The function

$$
(M_1, \ldots, M_n) \mapsto [\![\varphi]\!](\mathcal{A}, \alpha[X_1 \mapsto M_1, \ldots, X_n \mapsto M_n])
$$

is a measurable function from $\bigotimes_{i=1}^n \mathcal{B}(\mathcal{P}(A))$ to $\mathcal{B}(\mathbb{R})$.

Proof. We use induction on the structure of φ. For MSO formulas the claim is just the statement of Assumption 5.3. For constant functions, products and sums of measurable functions the statement follows from standard measure theory.

Let $\varphi = \mathbb{E}_p\, X.\varphi'$. By induction hypothesis, we know that the function f given by $f(N, M_1, \ldots, M_n) = [\![\varphi']\!](\mathcal{A}, \alpha[X_1 \mapsto M_1, \ldots, X_n \mapsto M_n][X \mapsto N])$ is measurable. We have

$$
[\![\varphi]\!](\mathcal{A}, \alpha[X_1 \mapsto M_1, \ldots, X_n \mapsto M_n]) = \int_{N\subseteq A} f(N, M_1, \ldots, M_n)\ \mathrm{Pr}_p^A(\mathrm{d}N),
$$

which is measurable by Fubini's theorem (Theorem 2.16). ∎

Though we only included conjunction and negation as Boolean connectives in the definition of probabilistic MSO, one can obtain the other operators as usual. We give two examples: let φ_1 and φ_2 be two PMSO formulas. We define the following abbreviations:

$$\varphi_1 \vee \varphi_2 = \neg((\neg\varphi_1) \wedge (\neg\varphi_2)), \qquad \text{and} \qquad \varphi_1 \to \varphi_2 = (\neg\varphi_1) \vee \varphi_2$$

The explicit semantics are:

$$[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] + [\![\varphi_2]\!] - [\![\varphi_1]\!][\![\varphi_2]\!] \quad \text{and} \quad [\![\varphi_1 \to \varphi_2]\!] = 1 - [\![\varphi_1]\!] + [\![\varphi_1]\!][\![\varphi_2]\!].$$

The semantics of the disjunction can be interpreted as probability: given two events A and B the probability of their union is $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$. If furthermore A and B are independent, we obtain $\Pr(A) + \Pr(B) - \Pr(A)\Pr(B)$, which has the same structure as $[\![\varphi_1 \vee \varphi_2]\!]$. For a MSO formula ψ and a probabilistic MSO formula φ, the semantics of $\psi \to \varphi$ selects the conclusion part of the implication only if the premise is true:

$$[\![\psi \to \varphi]\!](\mathcal{A}, \alpha) = \begin{cases} [\![\varphi]\!](\mathcal{A}, \alpha) & \text{if } (\mathcal{A}, \alpha) \models \psi \\ 1 & \text{otherwise.} \end{cases}$$

The probabilistic connectives satisfy many laws which one would expect from Boolean operators. We will give some equalities in the next lemma and state additional equalities regarding the expected value quantifier. Two probabilistic PMSO($\mathcal{S}$) formulas φ_1 and φ_2 are called *equivalent* if $[\![\varphi_1]\!](\mathcal{A}, \alpha) = [\![\varphi_2]\!](\mathcal{A}, \alpha)$ for all $\mathcal{S}$-structures $\mathcal{A}$ and $\mathcal{A}$-assignments α. In this case, we write $\varphi_1 \equiv \varphi_2$.

Some equivalences are only valid for a particular set of $\mathcal{S}$-structures. Let C be a set of $\mathcal{S}$-structures, we write $\varphi_1 \equiv_C \varphi_2$ if $[\![\varphi_1]\!](\mathcal{A}, \alpha) = [\![\varphi_2]\!](\mathcal{A}, \alpha)$ for all $\mathcal{A} \in C$ and $\mathcal{A}$-assignments α. In this case φ_1 and φ_2 are called *equivalent on* C.

Lemma 5.7. The following identities hold:

1. $\varphi_1 \wedge \varphi_2 \equiv \varphi_2 \wedge \varphi_1$ and $\varphi_1 \vee \varphi_2 \equiv \varphi_2 \vee \varphi_1$,
2. $(\varphi_1 \wedge \varphi_2) \wedge \varphi_3 \equiv \varphi_1 \wedge (\varphi_2 \wedge \varphi_3)$ and $(\varphi_1 \vee \varphi_2) \vee \varphi_3 \equiv \varphi_1 \vee (\varphi_2 \vee \varphi_3)$,
3. $\psi \vee (\varphi_1 \wedge \varphi_2) \equiv (\psi \vee \varphi_1) \wedge (\psi \vee \varphi_2)$ and $\psi \wedge (\varphi_1 \vee \varphi_2) \equiv (\psi \wedge \varphi_1) \vee (\psi \wedge \varphi_2)$
4. $\top \wedge \varphi \equiv \varphi$ and $\top \vee \varphi \equiv \top$,
5. $\bot \wedge \varphi \equiv \bot$ and $\bot \vee \varphi \equiv \varphi$,

6. $\neg\, \mathbb{E}_p\, X.\varphi \equiv \mathbb{E}_p\, X.\neg\varphi$,
7. $\varphi_1 \wedge \mathbb{E}_p\, X.\varphi_2 \equiv \mathbb{E}_p\, X.(\varphi_1 \wedge \varphi_2)$ if $X \notin \mathrm{Free}(\varphi_1)$,
8. $\mathbb{E}_p\, X.\varphi \equiv \varphi$ if $X \notin \mathrm{Free}(\varphi)$,
9. $\mathbb{E}_p\, X.\, \mathbb{E}_q\, Y.\varphi \equiv \mathbb{E}_q\, Y.\, \mathbb{E}_p\, X.\varphi$,

where φ, φ_1, φ_2 are probabilistic MSO formulas, ψ is a MSO formula, $\top$ is any formula with $[\![\top]\!] = 1$, and $\bot$ is any formula with $[\![\bot]\!] = 0$.

Note that distributivity does not hold in the general case of three probabilistic MSO formulas, but only if the factored out term is a MSO formula.

Proof. Statements 1 to 5 follow directly from the definition of the semantics. Statements 6,7 and 8 are a consequence of the linearity of the integral. Statement 9 is Fubini's theorem. ∎

Example 5.8. Let $\Sigma = \{\mathrm{a}, \mathrm{b}\}$ and consider the following $\mathrm{PMSO}(\mathcal{W}_\Sigma)$ formula φ:

$$\varphi = \mathbb{E}_p\, X.\forall x.(\mathrm{label}_{\mathrm{a}}(x) \implies x \in X).$$

We explicitly compute the semantics of φ. Let $w \in \Sigma^*$.

$$\begin{aligned}\|\varphi\|(w) &= \int_{M \subseteq \mathrm{pos}(w)} [\![\forall x.\, \mathrm{label}_{\mathrm{a}}(x) \implies x \in X]\!](w, \{X \mapsto M\})\ \mathrm{B}_p(\mathrm{d}M) \\ &= \mathrm{B}_p(\{M \subseteq \mathrm{pos}(w) \mid \{x \in \mathrm{pos}(w) \mid w_x = a\} \subseteq M\}) \\ &= p^{|w|_{\mathrm{a}}}.\end{aligned}$$

The last equation can be seen as follows: every a-labelled position must always be included in M with probability p. All other positions can or can not be included in M, thus, their probability sums up to 1.

Example 5.9. We return to the communication device from Examples 2.19 and 6.16. We give a $\mathrm{PMSO}(\mathcal{W}_\Sigma)$ formula φ with semantics $\|A\|$ from Example 2.19, i.e., $[\![\varphi]\!]$ is the probability that the sequence of wait and input events described by the word does not overflow the buffer.

$$\begin{aligned}\varphi = \mathbb{E}_p\, X.\, \mathbb{E}_q\, Y.\exists Z.(\exists x.(\forall y.x \leq y) \wedge x \notin Z) \wedge \forall x.\forall y.(y = x+1) \implies \\ \Big(\big((x \notin Z \wedge \mathrm{label}_{\mathrm{w}}(x)) \implies y \notin Z\big) \wedge \\ \big((x \notin Z \wedge \mathrm{label}_{\mathrm{i}}(x) \wedge x \in X) \implies y \notin Z\big) \wedge \\ \big((x \notin Z \wedge \mathrm{label}_{\mathrm{i}}(x) \wedge x \notin X) \implies y \in Z\big) \wedge \\ \big((x \in Z \wedge \mathrm{label}_{\mathrm{w}}(x) \wedge x \in Y) \implies y \notin Z\big) \wedge \\ \big((x \in Z \wedge \mathrm{label}_{\mathrm{w}}(x) \wedge x \notin Y) \implies y \in Z\big) \wedge \\ \big((x \in Z \wedge \mathrm{label}_{\mathrm{i}}(x)) \implies (y \in Z \wedge x \in X)\big)\Big)\end{aligned}$$

The set variables have the following meaning: X contains all positions where sending a newly incoming message is successful without previously storing the message in the buffer, Y contains all positions where sending a buffered message was successful, and Z contains all positions with full buffer. The second to the last line of the equation encode the transition conditions as explained in Example 2.19. Note the last line: if the buffer is full and a new message is received, the buffer is still full after this step and the newly received message must be sent successfully, since otherwise the buffer would overflow.

Example 5.10. In Examples 2.26 and 7.17 we considered the ranked alphabet Σ with $\Sigma_2 = \{\mathrm{f}\}$ and $\Sigma_0 = \{\mathrm{a}, \mathrm{b}\}$, and the tree series $S(t) = \sum_{x \in \mathrm{pos}_\mathrm{a}(t)} (1/2)^{|x|}$. We give a PMSO($\mathcal{T}_\Sigma$) formula φ with $[\![\varphi]\!] = S$.

$$\varphi = \mathbb{E}_{1/2} X. \exists x. \, \mathrm{label}_\mathrm{a}(x) \wedge \forall y. (y \neq x \wedge y \preceq x) \\ \implies \big(y \in X \iff (\exists z. \, \mathrm{edge}_2(y, z) \wedge z \preceq x)\big),$$

where $y \preceq x$ denotes the prefix relation. This relation can be modelled in MSO($\mathcal{T}_\Sigma$) by

$$(y \preceq x) = \Big(\forall X. \big(x \in X \wedge \forall u. \forall v. (v \in X \wedge (\mathrm{edge}_1(u, v) \vee \mathrm{edge}_2(u, v)) \\ \implies u \in X)\big) \implies y \in X\Big).$$

In φ the set X probabilistically chooses a leaf node by describing a path in the tree: if $x \notin X$ go left, otherwise go right. Thus, φ checks if the position at the end of the path described by X path is labelled by a.

As last result of this section, we want to derive a normal form for probabilistic MSO formulas, where all expected value quantifiers are in front of a Boolean MSO part and no probability constants occur. This normalisation process involves renaming of variables. This is easily possible in classical MSO logic and we show that this property carries over to probabilistic MSO logic.

Lemma 5.11. Let φ be a probabilistic MSO formula, $\mathcal{A}$ a $\mathcal{S}$-structure and α a $\mathcal{A}$-assignment. Furthermore, let $\mathfrak{X}$ and $\mathfrak{Y}$ be both first order or both second order variables. The following identity holds:

$$[\![\varphi]\!](\mathcal{A}, \alpha[\mathfrak{X} \mapsto \alpha(\mathfrak{Y})]) = [\![\varphi[\mathfrak{X} \leftarrow \mathfrak{Y}]]\!](\mathcal{A}, \alpha),$$

where $\varphi[\mathfrak{X} \leftarrow \mathfrak{Y}]$ is obtained from φ by replacing every *free* occurrence of $\mathfrak{X}$ by $\mathfrak{Y}$.

In particular, for a probabilistic MSO formula φ and a second-order variable Y that does not occur in φ, it holds that $\mathbb{E}_p X.\varphi = \mathbb{E}_p Y.\varphi[X \leftarrow Y]$.

Proof. The second statement is a direct consequence of the first statement and the definition of the semantics of $\mathbb{E}_p X$. We prove the first statement by induction on the structure of φ. For probability constants the statement is trivial. For MSO formulas the statement is a standard result. For conjunction, the induction hypothesis directly carries over:

$$\begin{aligned}[\![\varphi_1 \wedge \varphi_2]\!](\mathcal{A}, \alpha[\mathfrak{X} \mapsto \alpha(\mathfrak{Y})]) &= [\![\varphi_1]\!](\mathcal{A}, \alpha[\mathfrak{X} \mapsto \alpha(\mathfrak{Y})]) \cdot [\![\varphi_2]\!](\mathcal{A}, \alpha[\mathfrak{X} \mapsto \alpha(\mathfrak{Y})]) \\ &\overset{\text{IH}}{=} [\![\varphi_1[\mathfrak{X} \leftarrow \mathfrak{Y}]]\!](\mathcal{A}, \alpha) \cdot [\![\varphi_2[\mathfrak{X} \leftarrow \mathfrak{Y}]]\!](\mathcal{A}, \alpha) \\ &= [\![(\varphi_1 \wedge \varphi_2)[\mathfrak{X} \leftarrow \mathfrak{Y}]]\!](\mathcal{A}, \alpha).\end{aligned}$$

Negation is analogous to this case, and therefore omitted here.

Assume $\varphi = \mathbb{E}_p X.\varphi'$. If $X = \mathfrak{X}$, then $\mathfrak{X}$ is not free in φ. Hence, $\varphi[\mathfrak{X} \leftarrow \mathfrak{Y}] = \varphi$. Furthermore, the value of $\mathfrak{X}$ in $\alpha[\mathfrak{X} \mapsto \alpha(\mathfrak{Y})]$ is immediately overwritten by the application of $\mathbb{E}_p X$. Thus, the claim follows.

Assume $X \neq \mathfrak{X}$. We obtain

$$[\![\mathbb{E}_p X.\varphi']\!](\mathcal{A}, \alpha[\mathfrak{X} \mapsto \alpha(\mathfrak{Y})]) = \int [\![\varphi']\!](\mathcal{A}, \alpha[\mathfrak{X} \mapsto \alpha(\mathfrak{Y})][X \mapsto M])\ \mathrm{B}_p(\mathrm{d}M)$$

As $X \neq \mathfrak{X}$, we have $\alpha[\mathfrak{X} \mapsto \alpha(\mathfrak{Y})][X \mapsto M] = \alpha[X \mapsto M][\mathfrak{X} \mapsto \alpha[\mathfrak{Y}]$. This allows us to apply the induction hypothesis:

$$\begin{aligned}&= \int [\![\varphi'[\mathfrak{Y} \leftarrow \mathfrak{X}]]\!](\mathcal{A}, \alpha[X \mapsto M])\ \mathrm{B}_p(\mathrm{d}M) \\ &= [\![\mathbb{E}_p X.(\varphi'[\mathfrak{X} \leftarrow \mathfrak{Y}])]\!](\mathcal{A}, \alpha)\end{aligned}$$

As $\mathfrak{X} \neq X$, every occurence of $\mathfrak{X}$ is free in φ' if and only if it is free in φ.

$$= [\![\varphi[\mathfrak{X} \leftarrow \mathfrak{Y}]]\!](\mathcal{A}, \alpha)$$

This completes the proof. ∎

As second step towards a normal form for probabilistic MSO formulas, we want to eliminate probability constants. This is not possible for structures with an empty domain, as $\|\mathbb{E}_p X.\varphi\|(\mathcal{A}, \alpha) = \|\varphi\|(\mathcal{A}, \alpha[X \mapsto \emptyset])$ holds in this case. Thus, no true probability values can be introduced by the sole use of the expected value operator. The situation is different if the domain is non-empty. By fixing exactly one element of the domain in a set, one obtains exactly the probability p of $\mathbb{E}_p X$ as constant value. In the case of words one simply chooses the first position as fixed element. For trees the root position is definable. For arbitrary structures, it may be the case that no single position is definable. Thus, we assume that such a MSO formula exists.

Assumption 5.12. Let $\mathcal{S}$ be a signature and C a set of $\mathcal{S}$-structures. We say that C is *pointed* if there exists a MSO formula $\mathit{fix}(x)$ such that $\mathrm{Free}(\mathit{fix}(x)) = \{x\}$ and for every $\mathcal{A} \in C$ with $\mathrm{dom}(\mathcal{A}) \neq \emptyset$, there is an $a \in \mathrm{dom}(\mathcal{A})$ such that $\{a' \in \mathrm{dom}(\mathcal{A}) \mid (\mathcal{A}, \alpha[x \mapsto a']) \models \mathit{fix}(x)\} = \{a\}$ for all $\mathcal{A}$-assignments α.

The assumption may be violated in structures like bi-infinite words, where every position is essentially the same with respect to their order, c.f. [PP04]. As we are ultimately only interested in words and trees here, we assume that Assumption 5.12 holds for the rest of this chapter. This allows us to express probability constants using the expected value operator over non-empty domains.

Proposition 5.13. Let $\mathcal{S}$ be a signature and C a pointed set of $\mathcal{S}$-structures. Furthermore, let $\mathit{fix}(x)$ be the formula from Assumption 5.12. Then

$$[\![\mathbb{E}_p\, X.\exists x.x \in X \wedge \mathit{fix}(x)]\!](\mathcal{A}, \alpha) = p,$$

for all $\mathcal{A} \in C$ with $\mathrm{dom}(\mathcal{A}) \neq \emptyset$ and $\mathcal{A}$-assignments α.

Proof. Let $A = \mathrm{dom}(\mathcal{A})$ and $a \in A$ such that $\{a' \in A \mid (\mathcal{A}, \alpha[x \mapsto a']) \models \mathit{fix}(x)\} = \{a\}$ for all $\mathcal{A}$-assignments α. Thus, $(\mathcal{A}, \alpha) \models \exists x.x \in X \wedge \mathit{fix}(x)$ if and only if $a \in \alpha(X)$. This yields for every $\mathcal{A}$-assignment α that

$$[\![\mathbb{E}_p\, X.\exists x.x \in X \wedge \mathit{fix}(x)]\!](\mathcal{A}, \alpha) = \mathrm{B}_p(\{M \subseteq A \mid a \in M\}) = p. \qquad \blacksquare$$

We will now apply Lemmas 5.7 and 5.11 and Proposition 5.13 to transform every probabilistic MSO formula into a form where all expected value quantifiers are at the front of the formula and no probability constants occur any more. As the elimination of constants is only possible if the domain is non-empty, we can derive the normal form only if we add an explicit guard which checks if the domain is not empty.

Lemma 5.14. Let $\mathcal{S}$ be a signature and C be a pointed set of $\mathcal{S}$-structures. Furthermore, let φ be a probabilistic MSO formula. There are mutually distinct second order variables $X_1, \ldots, X_n$, probability values $p_1, \ldots, p_n$, and a MSO formula ψ such that

$$\varphi \wedge \eta \equiv_C \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.\psi,$$

where $\eta = (\exists x.x = x)$ is a check, whether the domain is empty.

Proof. We use induction on the structure of the formula. If φ is already a MSO formula, there is nothing to prove: $\varphi \wedge \eta$ is already in the claimed form. If $\varphi = p$, we apply Proposition 5.13. Note that this formula is 0 on structures with empty domain.

For $\varphi = \neg\varphi'$, assume $\varphi' \wedge \eta = \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.\psi'$. We apply Lemma 5.7 and obtain $\varphi \wedge \eta \equiv \neg(\varphi' \wedge \eta) \wedge \eta \equiv \neg(\mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.\psi') \wedge \eta \equiv \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.(\neg\psi' \wedge \eta)$.

In case $\varphi = \mathbb{E}_p X.\varphi'$, again assume $\varphi' \wedge \eta = \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.\psi'$. If $X = X_i$ for some $i = 1, \ldots, n$, then $[\![\varphi']\!]$ does not depend on the value of X. Thus, $\varphi \wedge \eta \equiv \varphi' \wedge \eta$. If $X \neq X_i$ for all $i = 1, \ldots, n$, we conclude $(\mathbb{E}_p X.\varphi') \wedge \eta = \mathbb{E}_p X.(\varphi' \wedge \eta) = \mathbb{E}_p X.\, \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.\psi'$.

Finally, assume $\varphi = \varphi_1 \wedge \varphi_2$. By induction hypothesis $\varphi_1 \wedge \eta \equiv \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.\psi_1$ and $\varphi_2 \wedge \eta \equiv \mathbb{E}_{q_1} Y_1.\, \mathbb{E}_{q_m} Y_m.\psi_2$ for some $p_1, \ldots, p_n, q_1, \ldots, q_m \in [0,1]$, $X_1, \ldots, X_n$, $Y_1, \ldots, Y_m \in \mathcal{V}_2$ and MSO($\mathcal{S}$) formulas ψ_1 and ψ_2. Using the last statement of Lemma 5.11 we may assume that the X_i's and Y_j's are pairwise distinct and that no X_i is free in ψ_2 and no Y_j is free in ψ_1 by renaming the quantified variables. By Lemma 5.7 we have

$$\varphi \wedge \eta \equiv (\varphi_1 \wedge \eta) \wedge (\varphi_2 \wedge \eta) \equiv \mathbb{E}_{p_1} X_1 \cdots \mathbb{E}_{p_n} X_n.\, \mathbb{E}_{q_1} Y_1 \cdots \mathbb{E}_{q_m} Y_m(\psi_1' \wedge \psi_2'). \qquad \blacksquare$$

5.3 Probabilistic Variants of First Order Quantifiers

The syntax of probabilistic MSO logic has been chosen quite minimal. Nevertheless, one can define additional logical operations as macros, i.e., they can be translated into probabilistic MSO as given in Definition 5.5.

5.3.1 Extended Universal First Order Quantifier

As in weighted logics, one can define an extended version of the first order universal quantifier to be applicable not only to Boolean formulas, but also to formulas that yield arbitrary values.

Definition 5.15. Let φ be a probabilistic MSO formula, we define the semantics of $\forall x.\varphi$ by

$$[\![\forall x.\varphi]\!](\mathcal{A}, \alpha) = \prod_{a \in \mathcal{A}} [\![\varphi]\!](\mathcal{A}, \alpha[x \mapsto a]),$$

for all $\mathcal{S}$-structures $\mathcal{A}$ and $\mathcal{A}$-assignments α.

A translation from this extended quantifier to probabilistic MSO is only possible if the quantified formula is of a simple form that we call step formula. It can be shown that, if $\forall x.\varphi$ is allowed for arbitrary probabilistic MSO formulas φ, the expressive power of PMSO is exceeded. This can be seen as follows: consider a

finite structure $\mathcal{A}$ and a PMSO sentence $\varphi = \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.\psi$ in normal form. Whenever $\|\varphi\|(\mathcal{A}) > 0$ holds, then $\|\varphi\|(\mathcal{A}) \geq (c_1 \cdots c_n)^{|\mathcal{A}|}$, where $c_i = \min(p_i, 1-p_i)$ for $i = 1, \ldots, n$. Thus, $\|\varphi\|$ decreases at most exponentially in the size of the structure. On the other hand, let $\lambda = \forall x.\forall y.p$. By definition, $\|\lambda\|(\mathcal{A}) = p^{|\mathcal{A}|^2}$. Hence, $\|\lambda\|$ decreases exponential in the square of the structure size. Therefore, $\|\lambda\|$ is not equivalent to the semantics of any probabilistic MSO sentence. This is the same as for weighted logics, where the same restriction is necessary to preserve recognizability of the formula's semantics.

Definition 5.16. We call a probabilistic MSO formula φ a *step formula* if it does not use the expected value operator $\mathbb{E}_p X$, i.e., φ is a Boolean combination of Boolean MSO formulas and probability constants.

Lemma 5.17. Let φ be a step formula. There is a probabilistic MSO formula η, where "$\forall x$." is only applied to MSO formulas, with $[\![\eta]\!] = [\![\forall x.\varphi]\!]$.

Proof. As φ is built using only MSO formulas, probability constants, conjunction and negation, there are MSO formulas $\psi_1, \ldots, \psi_n$ and probabilities $p_1, \ldots, p_n$ such that

$$\varphi \equiv \bigwedge_{i=1}^{n} (\psi_i \implies p_i).$$

We define the formula η by

$$\eta = \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.\forall x. \bigwedge_{i=1}^{n} (\psi_i \implies x \in X_i),$$

where the second-order variables $X_1, \ldots, X_n$ are new variables not occurring in φ. We show $[\![\eta]\!] = [\![\forall x.\varphi]\!]$: let $\mathcal{A}$ be a $\mathcal{S}$-structure and α be an $\mathcal{A}$-assignment. We define $\Psi_i = \{ a \in A \mid (\mathcal{A}, \alpha[x \mapsto a]) \models \psi_i \}$ for every $i = 1, \ldots, n$. We compute

$$[\![\eta]\!](\mathcal{A}, \alpha) = \int \mathrm{B}_{p_1}(\mathrm{d}M_1) \cdots \int \mathrm{B}_{p_n}(\mathrm{d}M_n) \prod_{i=1}^{n} \begin{cases} 1 & \text{if } a \in M_i \text{ for all } a \in A \text{ with } (\mathcal{A}, \alpha[x \mapsto a]) \models \psi_i \\ 0 & \text{otherwise} \end{cases}$$

By linearity of the integral, we can move the product in front of every integral:

$$= \prod_{i=1}^{n} \int \mathrm{B}_{p_i}(\mathrm{d}M) \begin{cases} 1 & \text{if } \Psi_i \subseteq M_i \\ 0 & \text{otherwise} \end{cases}$$

$$= \prod_{i=1}^{n} \mathrm{B}_{p_i}(\{ M \mid \Psi_i \subseteq M \})$$

We postpone the proof of the next equality to the end of the proof.

$$
\begin{aligned}
&= \prod_{i=1}^{n} p_i^{|\Psi_i|} && (*)\\
&= \prod_{i=1}^{n} \prod_{a \in A} \begin{cases} p_i & \text{if } k \in (\mathcal{A}, \alpha[x \mapsto a]) \models \psi_i \\ 1 & \text{otherwise} \end{cases} \\
&= \prod_{a \in A} [\![\varphi]\!](\mathcal{A}, \alpha[x \mapsto a]) \\
&= [\![\forall x.\varphi]\!](\mathcal{A}, \alpha).
\end{aligned}
$$

Note that we can rearrange the (possibly infinite) products as only real numbers from the interval $[0, 1]$ occur: a product $\prod_{i=1}^{\infty} \lambda_i$ converges to λ if and only if $\sum_{i=1}^{\infty} -\log(\lambda_i)$ converges to $-\log(\lambda)$. Here, all summands in this sum are non-negative reals, hence if the series is convergent it is absolute convergent and therefore unconditionally convergent. If $\lambda = 0$, the series converges to $+\infty$ and so does every rearrangement.

We still need to show $(*)$, i.e., $\mathrm{B}_p(\{M \mid X \subseteq M\}) = p^{|X|}$. For infinite sets X, we use the usual convention that $p^\infty = 0$ if $p < 1$ and $p^\infty = 1$ if $p = 1$. Let $A = \mathrm{dom}(\mathcal{A})$ and fix an enumeration $E = (a_1, a_2, \ldots)$ of A. Let $X \subseteq A$. We conclude by the continuity of measures:

$$
\begin{aligned}
\mathrm{B}_p(\{M \mid X \subseteq M\}) &= \lim_{n \to \infty} \mathrm{B}_{p,E}(\{M \mid X \cap \{a_1, \ldots, a_n\} \subseteq M\}) \\
&= \lim_{n \to \infty} \sum_{\substack{M \subseteq \{a_1, \ldots, a_n\} \\ X \cap \{a_1, \ldots, a_n\} \subseteq M}} \mathrm{B}_{p,E}(\mathrm{Cyl}_E^n(M)) \\
&= \lim_{n \to \infty} \sum_{\substack{M \subseteq \{a_1, \ldots, a_n\} \\ X \cap \{a_1, \ldots, a_n\} \subseteq M}} p^{|M|}(1-p)^{n-|M|} \\
&= \lim_{n \to \infty} p^{|X \cap \{a_1, \ldots, a_n\}|} = p^{|X|}.
\end{aligned}
$$

Hence, by application of the above transformation to every occurrence of $\forall x.\eta$ in a probabilistic MSO formula, we can obtain a new probabilistic MSO formula, where $\forall x$ is only applied to MSO formulas. ∎

5.3.2 First Order Expected Value Quantifier

In Definition 5.5 we only gave an expected value operator for second order variables. We give a first order expected value operator in this section. Whereas the stochastic process behind $\mathbb{E}_p X$ was to toss a coin for every position of the domain, we use

the notion of probability of first success for $\mathbb{E}_p\, x$. Fix some well order $\sqsubseteq$ on the domain. We consider the first element, with respect to $\sqsubseteq$, and toss an unfair coin. With probability p the process stops and is successful at this element. Otherwise, with probability $1-p$, the process moves on to the next element of the domain and starts over. This is also the model of the geometric distribution on $\mathbb{N}$.

Definition 5.18. Let $\mathcal{A}$ be an $\mathcal{S}$-structure and $\sqsubseteq$ a well order on $\mathcal{A}$. We define the semantics of the formula $\mathbb{E}_p\, x.\varphi$ for any PMSO($\mathcal{S}$) formula by

$$[\![\mathbb{E}_p\, x.\varphi]\!](\mathcal{A},\alpha) = \sum_{a\in \mathrm{dom}(\mathcal{A})} [\![\varphi]\!](\mathcal{A}, \alpha[x\mapsto a]) \cdot p(1-p)^{N_a},$$

where $N_a = |\{\, a' \in \mathcal{A} \mid a' \sqsubseteq a,\ a \neq a' \}| \in \mathbb{N} \cup \{\infty\}$.

Like with the extended universal first order quantifier, this operator does not add any expressive to probabilistic MSO logic. It can be translated to the syntax of Definition 5.5. Whereas the well order $\sqsubseteq$ is inherent to the definition of $\mathbb{E}_p\, x.\varphi$, it must be definable in MSO($\mathcal{S}$) to obtain a PMSO($\mathcal{S}$) formula equivalent to $\mathbb{E}_p\, x.\varphi$.

We say $\sqsubseteq$ is *MSO definable* over a set of $\mathcal{S}$-structures C if there is a MSO formula $\tau(x,y)$ with $\mathrm{Free}(\tau(x,y)) = \{x,y\}$ such that $a \sqsubseteq a'$ holds if and only if $(\mathcal{A}, \{x\mapsto a, y\mapsto a'\}) \models \tau(x,y)$ for all $a, a' \in \mathcal{A}$ and $\mathcal{A} \in C$.

For finite or infinite words one could use the natural order on the set of positions. On finite trees, the depth first search order is an example of a MSO definable linear order.

Lemma 5.19. Let $\sqsubseteq$ be a MSO definable well order over some set of $\mathcal{S}$-structures C. Let φ be a probabilistic MSO formula, $x \in \mathcal{V}_1$ and $p \in [0,1]$. There is a probabilistic MSO formula φ' with $\|\mathbb{E}_p\, x.\varphi\|(\mathcal{A},\alpha) = \|\varphi'\|(\mathcal{A},\alpha)$ for all $\mathcal{A} \in C$ and $\mathcal{A}$-assignments α.

Proof. Let τ be the MSO formula modelling $\sqsubseteq$ as described below Definition 5.18. We define the formula φ' as

$$\varphi' = \mathbb{E}_p\, X.(\widetilde{\varphi} \wedge \exists x.x \in X),$$

where X is a new variable symbol not in φ, and $\widetilde{\varphi}$ arises from φ by replacing every occurrence of $R(x_1,\dots,x_n)$ for any $R \in \mathcal{S}$ and $x_1,\dots,x_n \in \mathcal{V}_1$ with

$$\exists \widetilde{x_1}.\cdots\exists \widetilde{x_n}.R(\widetilde{x_1},\dots,\widetilde{x_n}) \wedge \bigwedge_{i=1}^{n} \begin{cases} \widetilde{x_i} = x_i & \text{if } x_i \neq x \\ \widetilde{x_i} \in X \wedge \forall y.y \in X \to \tau(\widetilde{x_i}, y) & \text{if } x_i = x, \end{cases}$$

where $\widetilde{x_1},\dots,\widetilde{x_n}, y$ are new variable symbols. Formulas of the form $x \in X$ and $x_1 = x_2$ are replaced in the same way. Using structural induction one shows that

$\|\widetilde{\varphi}\|(\mathcal{A},\alpha) = \|\varphi\|(\mathcal{A},\alpha[x \mapsto \min_{\sqsubseteq}(\alpha(X))])$ for all $\mathcal{A}$-assignment α with $\alpha(X) \neq \emptyset$, where $\min_{\sqsubseteq}(M)$ for a set $\emptyset \neq M \subseteq \mathcal{A}$ denotes the minimal element with respect to $\sqsubseteq$. This element always exists as $\sqsubseteq$ is a well order.

We show $\|\varphi'\| = \|\mathbb{E}_p\, x.\varphi\|$. Let $\mathcal{A} \in C$ and α an $\mathcal{A}$-assignment. We obtain

$$\begin{aligned}\|\varphi'\|(\mathcal{A},\alpha) &= \int_{M\neq\emptyset} [\![\widetilde{\varphi}]\!](\mathcal{A},\alpha[X \mapsto M])\ \mathrm{B}_p(\mathrm{d}M)\\ &= \sum_{a\in A}\int_{M\neq\emptyset,\,\min(M)=a} \|\varphi\|(\mathcal{A},\alpha[x\mapsto\min(M)])\ \mathrm{B}_p(\mathrm{d}M)\\ &= \sum_{a\in A}\|\varphi\|(\mathcal{A},\alpha[x\mapsto a])\,\mathrm{B}_p(\{M \mid M\neq\emptyset,\ \min(M)=a\}).\end{aligned}$$

Thus, we need to show $\mathrm{B}_p(\{M \mid M \neq \emptyset,\ \min(M) = a\}) = p(1-p)^{N_a}$ to complete the proof, where $N_a = |\{a' \in \mathcal{A} \mid a' \sqsubseteq a,\ a' \neq a\}|$. If $p = 0$ the whole probability mass is concentrated in $\{\emptyset\}$. Thus, $\mathrm{B}_p(\{M \mid M \neq \emptyset\}) = 0$ and the equation is satisfied.

Assume $p > 0$. Let $E = (a_1, a_2, \dots)$ an enumeration of A with $a_i \sqsubseteq a_{i+1}$ for all $i \geq 1$. In case $N_a = k < \infty$, we have $a_{k+1} = a$. We obtain

$$\mathrm{B}_p(\{M \neq \emptyset \mid \min(M) = a\}) = \mathrm{B}_p(\mathrm{Cyl}_E^{k+1}(\{a\})) = p(1-p)^k.$$

If $N_a = \infty$, there are infinitely many elements less than a. As $1-p < 1$, we conclude

$$\begin{aligned}\mathrm{B}_p(\{M \mid M\neq\emptyset,\ \min_{\sqsubseteq}(M)=a\}) &\leq \lim_{n\to\infty}\mathrm{B}_p(\{M \mid M\cap\{a_1,\dots,a_n\}=\emptyset\})\\ &= \lim_{n\to\infty}\mathrm{B}_p(\mathrm{Cyl}_E^n(\emptyset))\\ &= \lim_{n\to\infty}(1-p)^n = 0.\end{aligned}$$

Since $(1-p)^{N_a} = (1-p)^\infty = 0$, the proof is complete. ∎

5.4 Equivalence to Nivat classes

In this section, we give the proof that probabilistic MSO logic is equally expressive as probabilistic Nivat classes, and therefore, also equally expressive as probabilistic Muller-automata, probabilistic bottom-up tree automata, respectively.

Before we can show this statement, we need two preparatory results. In the first result, we decompose Bernoulli measure over an arbitrary finite set into a product of binary Bernoulli measures. The second result states that we can switch between words/trees of tuples and tuples of words/trees without changing the probability. This is due to the independence of different positions in Bernoulli measures.

Lemma 5.20. Let M be a finite set and d a distribution on M. There is a number $n \geq 1$, probability values $p_1, \ldots, p_n \in [0,1]$, and a function $f\colon \{0,1\}^n \to M$ such that $d = (\bigotimes_{i=1}^n d_i) \circ f^{-1}$, where d_i is a distribution on $\{0,1\}$ with $d_i(1) = p_i$ and $d_i(0) = 1 - p_i$ for all $i = 1, \ldots, n$.

Proof. Let $M = \{a_1, \ldots, a_m\}$. Define $n = m - 1$ the values $p_1, \ldots, p_n$ by

$$p_i = \frac{d(a_i)}{1 - \sum_{j=1}^{i-1} d(a_j)}.$$

As $\sum_{i=1}^n d(a_i) = 1$, we have $p_i \leq 1$ for all $i = 1, \ldots, n$. The function f is given by

$$f(x_1, \ldots, x_n) = \begin{cases} a_k & \text{if } k = \min\{i \mid x_i = 1\} \text{ and } \{i \mid x_i = 1\} \neq \emptyset \\ a_m & \text{if } x_i = 0 \text{ for all } i = 1, \ldots, n. \end{cases}$$

Before we show $\mathrm{B}_d = (\bigotimes_{i=1}^n d_i) \circ f^{-1}$, we prove that $\prod_{j=1}^i (1 - p_j) = 1 - \sum_{j=1}^i d(a_j)$ via induction over i. For $i = 1$, the statement is clear. Assume the statement holds for some i. We obtain for $i + 1$:

$$\prod_{j=1}^{i+1} (1 - p_j) = \left(1 - \sum_{j=1}^{i} d(a_j)\right)\left(1 - \frac{d(a_{i+1})}{1 - \sum_{j=1}^i d(a_j)}\right) = 1 - \sum_{j=1}^{i+1} d(a_j).$$

Let $a_k \in M$. First consider the case $k < m$. By definition of f we have $f^{-1}(\{a_k\}) = \{(x_1, \ldots, x_n) \in M \mid x_1 = \cdots = x_{k-1} = 0,\ x_k = 1\}$. Thus,

$$\left(\bigotimes_{i=1}^n d_i\right)(f^{-1}(\{a_k\})) = p_k \prod_{i=1}^{k-1} (1 - p_i) = \frac{d(a_k)}{1 - \sum_{i=1}^{k-1} d(a_i)}\left(1 - \sum_{i=1}^{k-1} d(a_i)\right) = d(a_k).$$

For $k = m$, we have $f^{-1}(\{a_k\}) = \{(0, \ldots, 0)\}$. Therefore, $(\bigotimes_{i=1}^n d_i)(f^{-1}(\{a_m\}) = \prod_{i=1}^n (1 - p_i) = 1 - \sum_{i=1}^{m-1} d(a_i) = d(a_m)$. This shows that the distributions are equal. ∎

Proposition 5.21. Let $n \geq 1$ and $N, M_1, \ldots, M_n$ be finite sets, d a distribution on N, d_i a distribution on M_i for $i = 1, \ldots, n$. Furthermore, let $\overline{M} = M_1 \times \cdots \times M_n$, $\overline{d} = d_1 \otimes \cdots \otimes d_n$, and $f\colon \overline{M} \to N$ be a function such that $d = \overline{d} \circ f^{-1}$. Then the following statements hold:

1. Let $k \in \mathbb{N} \cup \{\omega\}$. Then $(\mathrm{B}_{d_1}^k \otimes \cdots \otimes \mathrm{B}_{d_n}^k) \circ \left(\widetilde{f}\right)^{-1} = \mathrm{B}_d^k$ on $\mathcal{B}(N^n)$, where $\widetilde{f}\colon M_1^k \times \cdots \times M_n^k \to N^k$ is given by $\widetilde{f}(u^{(1)}, \ldots, u^{(n)}) = (f(u_i^{(1)}, \ldots, u_i^{(n)}))_{i=1}^k$ where $u^{(i)} = (u_j^{(i)})_{j=1}^k$ for $i = 1, \ldots, n$.

2. Let D be a finite, non-empty set. Then $(\mathrm{B}_{d_1}^D \otimes \cdots \otimes \mathrm{B}_{d_n}^D) \circ \left(\widetilde{f}\right)^{-1} = \mathrm{B}_d^D$ on $\mathcal{B}(N^D)$, where $\widetilde{f}\colon M_1^D \times \cdots \times M_n^D \to N^D$ is given by $\widetilde{f}(t_1, \ldots, t_n) = \big(f(t_1(x), \ldots, t_n(x))\big)_{x \in D}$ for all $t_i \in M_i^D$ where $i = 1, \ldots, n$.

Note that if we choose $N = M_1 \times \cdots \times M_n$ and $f = \mathrm{id}_N$ in Proposition 5.21, we obtain that we can switch from tuples of words/trees in the product space to words/trees of tuples in the Borel space over tuples of letters.

Proof. We only show the infinite word case of 1., the proof of the finite word case and the proof of 2. are analogous. Let $A = w_1 \cdots w_\ell N^\omega$ be a cylinder set in $\mathcal{B}(M^\omega)$. Then, the preimage of A is given by

$$\widetilde{f}^{-1}(A) = \bigcup_{\substack{u_1^{(i)}, \ldots, u_\ell^{(i)} \in M_i \ (i=1,\ldots,n) \\ f(u_j^{(1)}, \ldots, u_j^{(n)}) = w_j \text{ for all } j \geq 1}} \bigtimes_{i=1}^{n} u_1^{(i)} \cdots u_\ell^{(i)} M_i^\omega,$$

where $\bigtimes_{i=1}^n$ is the n-ary Cartesian product. Let $\mu = \mathrm{B}_{d_1}^\omega \otimes \cdots \otimes \mathrm{B}_{d_n}^\omega$. We obtain

$$\begin{aligned}
\mu\big(\widetilde{f}^{-1}(A)\big) &= \sum_{\substack{u_1^{(i)}, \ldots, u_\ell^{(i)} \in M_i \ (i=1,\ldots,n) \\ f(u_j^{(1)}, \ldots, u_j^{(n)}) = w_j \text{ for all } j=1,\ldots,\ell}} \prod_{i=1}^{n} \prod_{j=1}^{\ell} d_j(u_j^{(i)}) \\
&= \prod_{j=1}^{\ell} \sum_{\substack{u^{(i)} \in M_i \ (i=1,\ldots,n) \\ f(u^{(1)}, \ldots, u^{(n)}) = w_i}} \prod_{i=1}^{n} d_j(u^{(i)}) \\
&= \prod_{i=1}^{\ell} \overline{d}(f^{-1}(\{w_i\})) = \prod_{i=1}^{\ell} d(w_i) \\
&= \mathrm{B}_d(w_1 \cdots w_\ell N^\omega) = \mathrm{B}_d(A).
\end{aligned}$$

Therefore, $\left(\mathrm{B}_{d_1}^\omega \otimes \cdots \otimes \mathrm{B}_{d_n}^\omega\right) \circ \left(\widetilde{f}\right)^{-1} = \mathrm{B}_d^\omega$ on $\mathcal{B}(N^\omega)$ as claimed. ∎

We are now ready to state and prove the two main results of this chapter: the expressive equivalence of probabilistic MSO logic and Nivat representations for words and finite trees. We will state these results as two separate theorems. As the proofs only depends in small parts on the actual choice of the structure, we will prove the follows two theorems together.

Theorem 5.22. Let Σ be a finite alphabet and $S\colon \Sigma^\infty \to [0,1]$ be any function. The following statements are equivalent:

1. $S = [\![\varphi]\!]$ for a probabilistic MSO sentence $\varphi \in \mathrm{PMSO}(\mathcal{W}_\Sigma)$,
2. $S_+ \in \mathcal{N}(\Sigma^\infty)$,

where $S_+(w) = S(w)$ if $w \neq \varepsilon$ and $S_+(\varepsilon) = 0$. The translations are effective in both directions.

Theorem 5.23. Let Σ be a finite ranked alphabet and $S\colon \mathrm{T}_\Sigma \to [0,1]$ be any function. The following statements are equivalent:

1. $S = [\![\varphi]\!]$ for a probabilistic MSO sentence $\varphi \in \mathrm{PMSO}(\mathcal{T}_\Sigma)$,
2. $S \in \mathcal{N}(\mathrm{T}_\Sigma)$.

The translations are effective in both directions.

In order to get a unified representation for words and finite trees let $\mathcal{S} = \mathcal{W}_\Sigma$ or $\mathcal{S} = \mathcal{T}_\Sigma$. We define for every $\mathcal{S}$-structure $\mathcal{A}$ and function $f\colon \Sigma \to \Gamma$, the image of $\mathcal{A}$ by $f(\mathcal{A}) = (\mathrm{dom}(\mathcal{A}), (R^{f(\mathcal{A})})_{R\in\mathcal{S}})$ where $\mathrm{label}_a^{f(\mathcal{A})} = \bigcup_{a'\in\Sigma,\, f(a')=a} \mathrm{label}_{a'}^{\mathcal{A}}$ and $R^{f(\mathcal{A})} = R^{\mathcal{A}}$ if $R \neq \mathrm{label}_a$ for some $a \in \Gamma$.

This definition corresponds to the homomorphic image of words and the image under relabellings of trees, i.e. $\widetilde{f(w)} = \widetilde{f(w)}$ and $f(\widetilde{t}) = \widetilde{f(t)}$ for all $w \in \Sigma^\infty$ and $t \in \mathrm{T}_\Sigma$, where $\widetilde{w}$, $\widetilde{t}$, $\widetilde{f(w)}$ and $\widetilde{f(t)}$ are the structures introduced in Section 4.1.

Proof. The proof relies only in small parts on the actual choice of C. Paragraphs that are only valid for words or trees are marked with **w.** or **t.**, respectively.

Let $S = [\![\varphi]\!]_C$ for a probabilistic MSO sentence $\varphi \in \mathrm{PMSO}(\mathcal{S})$. By Lemma 5.14 we may assume that $\varphi_1 = \varphi \wedge \eta$, where $\eta = \exists x.x = x$, is of the form

$$\varphi_1 = \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n.\psi,$$

for probability values $p_1, \ldots, p_n \in [0,1]$, second order variables $X_1, \ldots, X_n$, and a Boolean MSO formula ψ. Note that $\|\varphi_1\| = S_+$. We show $[\![\varphi_1]\!] \in \mathcal{N}(C)$:

$$[\![\varphi_1]\!](\mathcal{A}) = \int \cdots \int [\![\psi]\!](\mathcal{A}, \{X_1 \mapsto M_1, \ldots, X_n \mapsto M_n\})\ \mathrm{B}_{p_1}(\mathrm{d}M_1) \cdots \mathrm{B}_{p_n}(\mathrm{d}M_n)$$

By Fubini's theorem, we can change the iterated integration to a single integral over the product space:

$$= \int [\![\psi]\!](\mathcal{A}, \{X_1 \mapsto M_1, \ldots, X_n \mapsto M_n\})(\mathrm{B}_{p_1} \otimes \cdots \otimes \mathrm{B}_{p_n})(\mathrm{d}(M_1, \ldots, M_n))$$

As $[\![\varphi]\!]$ attains only values in $\{0,1\}$ the integral is just the measure of a set:

$$= \left(\bigotimes_{i=1}^{n} \mathrm{B}_{p_i}\right)\big(\{(M_1,\dots,M_n) \in \mathcal{P}(\mathrm{dom}(A))^n \mid (\mathcal{A}, \{X_1 \mapsto M_1, \dots, X_n \mapsto M_n\}) \models \psi\}\big)$$

Using the measurable mappings $\mathcal{P}(\mathrm{dom}(\mathcal{A}))^n \to \left(\{0,1\}^{\mathrm{dom}(\mathcal{A})}\right)^n \to (\{0,1\}^n)^{\mathrm{dom}(\mathcal{A})}$ we obtain, by Lemma 5.2 and Proposition 5.21, a single Bernoulli measure:

$$= \mathrm{B}_d\ \big(\{(u_1,\dots,u_n) \in M^{\mathrm{dom}(\mathcal{A})} \mid (\mathcal{A}, \{X_1 \mapsto \mathrm{supp}(u_1), \dots, X_n \mapsto \mathrm{supp}(u_n)\}) \models \psi\}), \qquad (*)$$

where we set $M = \{0,1\}^n$ and the distribution d on M is given by $d(a_1,\dots,a_n) = \prod_{i=1}^{n}(a_i p_i + (1-a_i)(1-p_i))$.

w. In the case of words, consider the alphabet $\Gamma = \Sigma \times M$ and the language

$$L = \{(w, u_1, \dots, u_n) \in \Gamma^\infty \mid (w, \{X_1 \mapsto \mathrm{supp}(u_1), \dots, X_n \mapsto \mathrm{supp}(u_n)\} \models \psi\}.$$

By (the proof of) Büchi's theorem, L is a regular language. Moreover, we have $\mathrm{B}_d(\{u \in M^\infty \mid (w,u) \in L\}) = (*)$. By setting $g\colon \Gamma^\infty \to M^\infty$ and $h\colon \Gamma^\infty \to \Sigma^\infty$ the canonical projections, we see that $S_+(w) = [\![\varphi_1]\!] = (\mathrm{B}_d \circ g)(h^{-1}(\{w\}) \cap L)$. We conclude $S_+ \in \mathcal{N}(\Sigma^\infty)$.

t. Now, consider the case that $C = \mathrm{T}_\Sigma$. We define the ranked alphabet $\Gamma = \Sigma \times M$, i.e., $\mathit{arity}_\Gamma((f,m)) = \mathit{arity}_\Sigma(f)$ for all $(f,m) \in \Gamma$. Again, we consider the tree language

$$L = \{(t, u_1, \dots, u_n) \in \mathrm{T}_\Gamma \mid (t, \{X_1 \mapsto \mathrm{supp}(u_1), \dots, X_n \mapsto \mathrm{supp}(u_n)\} \models \psi\}.$$

By (the proof of) Theorem 4.7, we obtain that L is a regular tree language. Thus, by letting $g\colon \mathrm{T}_\Gamma \to \mathrm{T}_M$ and $h\colon \mathrm{T}_\Gamma \to \mathrm{T}_\Sigma$ be the canonical projections, we obtain, as in the word case, $S_+ \in \mathcal{N}(\mathrm{T}_\Sigma)$.

Conversely, assume $S_+ \in \mathcal{N}(C)$. Let Γ, M, d, g, h, L as in Definition 3.6, Definition 3.12 respectively. By Büchi's theorem, there is a MSO sentence $\psi \in \mathrm{MSO}(\mathcal{W}_\Gamma)$, $\psi \in \mathrm{MSO}(\mathcal{T}_\Gamma)$ respectively, such that $L = \mathrm{L}_C(\psi)$. Assume $\Gamma = \{a_1,\dots,a_m\}$ and let $Y_1,\dots,Y_m$ be new second order variables. We transform ψ into a formula $\widetilde{\psi}$ by replacing every occurrence of $\mathrm{label}_{a_i}(x)$ with $x \in Y_i$. Then, $\widetilde{\psi}$ does not contain any atomic formulas of the form label_a. Thus, we can regard $\widetilde{\psi}$ as a MSO formula over Σ. Using structural induction one shows

$$(\mathcal{A}', \alpha[Y_i \mapsto \mathrm{label}_{a_i}^{\mathcal{A}}]_{i=1}^m) \models \widetilde{\psi} \iff (\mathcal{A}, \alpha) \models \psi, \qquad (5.2)$$

where $\mathcal{A}$ is a $\mathcal{W}_\Gamma$-structure, $\mathcal{T}_\Gamma$-structure respectively, and $\mathcal{A}'$ is a $\mathcal{W}_\Sigma$-structure, $\mathcal{T}_\Sigma$-structure respectively, such that $\mathcal{A}$ and $\mathcal{A}'$ only differ in their label_a relations.

By Lemma 5.20 there is a number $n \geq 1$, probabilities $p_1, \ldots, p_n$ and a function $f\colon \{0,1\}^n \to M$ such that $d = (\bigotimes_{i=1}^n d_i) \circ f^{-1}$, where $d_i \in \Delta(\{0,1\})$ with $d_i(1) = p_i$. Let $X_1, \ldots, X_n$ be new variables. We define a probabilistic MSO sentence φ_1 over $\mathcal{S}$, where $\mathit{part}(Y_1, \ldots, Y_n)$ is a MSO sentence stating that $Y_1, \ldots, Y_n$ are a partition of the domain:

$$\begin{aligned}
\varphi_1 = \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n. \exists Y_1. \cdots \exists Y_m. & \\
& \wedge \mathit{part}(Y_1, \ldots, Y_m) \wedge \forall x. \bigwedge_{a_i \in \Gamma} x \in Y_i \implies \mathrm{label}_{h(a_i)}(x) && (5.3)\\
& \wedge \widetilde{\psi} && (5.4)\\
& \wedge \forall x. \bigwedge_{a_i \in \Gamma} x \in Y_i \implies \bigvee_{\substack{(x_1,\ldots,x_n)\in\{0,1\}^n \\ f(x_1,\ldots,x_n)=g(a_i)}} \begin{cases} x \in X_i & \text{if } x_i = 1 \\ x \notin X_i & \text{if } x_i = 0. \end{cases} && (5.5)
\end{aligned}$$

Let ψ_1 be the Boolean part of φ_1, i.e. from (5.3) to (5.5). If $\mathcal{S} = \mathcal{W}_\Sigma$, set $\mathcal{S}' = \mathcal{W}_\Gamma$. Otherwise, if $\mathcal{S} = \mathcal{T}_\Sigma$, let $\mathcal{S}' = \mathcal{T}_\Gamma$. Let $\mathcal{A}$ be a $\mathcal{S}$ structure and α an $\mathcal{A}$-assignment. We show that $(\mathcal{A}, \alpha) \models \exists Y_1. \cdots \exists Y_m. \psi_1$ if and only if there is a $\mathcal{S}'$-structure $\mathcal{A}'$ such that the label relations of $\mathcal{A}'$ are a partition of the domain, $\mathcal{A}' \models \psi$, $h(\mathcal{A}') = \mathcal{A}$ and $f(\mathbb{1}_{\alpha(X_1)}(x), \ldots, \mathbb{1}_{\alpha(X_n)}(x)) = g(a)$ for all $x \in \mathrm{label}_a^{\mathcal{A}'}$ and $a \in \Gamma$.

Assume there is a $\mathcal{S}$-structure $\mathcal{A} \in C$ and a $\mathcal{A}$-assignment α such that $(\mathcal{A}, \alpha) \models \psi_1$. Let $M_{a_i} = \alpha(Y_{a_i})$ for $i = 1, \ldots, m$. By (5.3) we derive that $(M_a)_{a\in\Gamma}$ is a partition of $\mathrm{dom}(\mathcal{A})$. Moreover, $M_a \subseteq \mathrm{label}_{h(a)}^{\mathcal{A}}$ holds for all $a \in \Gamma$. Define $\mathcal{A}' = (\mathrm{dom}(\mathcal{A}), (R^{\mathcal{A}'})_{R\in\mathcal{S}'})$ with $\mathrm{label}_a^{\mathcal{A}'} = M_a$ for all $a \in \Gamma$ and $R^{\mathcal{A}'} = R^{\mathcal{A}}$ for all $R \in \mathcal{S}'$ with $R \neq \mathrm{label}_a$ for all $a \in \Gamma$. From (5.3) we conclude $h(\mathcal{A}') = \mathcal{A}$. By (5.2) we have $\mathcal{A}' \models \psi$. Finally, $f(\mathbb{1}_{\alpha(X_1)}(x), \ldots, \mathbb{1}_{\alpha(X_n)}(x)) = g(a)$ for all $x \in \mathrm{label}_a^{\mathcal{A}'}$ and $a \in \Gamma$ is just the statement of (5.5).

Conversely, assume there is a $\mathcal{W}_\Gamma$-structure, $\mathcal{T}_\Gamma$-structure respectively, $\mathcal{A}'$ such that the label relations in $\mathcal{A}'$ partition the domain, $\mathcal{A}' \models \psi$, $h(\mathcal{A}') = \mathcal{A}$, and $f(\mathbb{1}_{\alpha(X_1)}(x), \ldots, \mathbb{1}_{\alpha(X_n)}(x)) = g(a)$ for all $a \in \Gamma$ and $x \in \mathrm{label}_a^{\mathcal{A}'}$. By defining sets $M_i = \mathrm{label}_{a_i}^{\mathcal{A}'}$ we conclude $(\mathcal{A}, \alpha[Y_i \mapsto M_i]_{i=1}^m) \models \psi_1$ directly from the definition of ψ_1 and using (5.2). Thus, $(\mathcal{A}, \alpha) \models \exists Y_1. \cdots \exists Y_m. \psi_1$.

Using this correspondence we obtain for the semantics of φ_1:

$$[\![\varphi_1]\!](\mathcal{A}) = \int \cdots \int [\![\exists Y_1. \cdots \exists Y_m. \psi_1]\!](\mathcal{A}, \alpha[X_i \mapsto M_i]_{i=1}^n)\ \mathrm{B}_{p_1}(\mathrm{d}M_1) \cdots \mathrm{B}_{p_n}(\mathrm{d}M_n)$$

We apply Fubini's theorem to switch to the product space:

$$= \int [\![\exists Y_1. \cdots \exists Y_m. \psi_1]\!](\mathcal{A}, \alpha[X_i \mapsto P_i]_{i=1}^n)\ (\textstyle\bigotimes_{i=1}^n \mathrm{B}_{p_i})(\mathrm{d}(P_1, \ldots, P_n))$$

The integrated function attains only values in $\{0,1\}$, thus, the integral is just the measure of a set:

$$= \left(\bigotimes_{i=1}^{n} \mathrm{B}_{p_i}\right)\big(\{(M_1,\dots,M_n) \in \mathcal{P}(\mathrm{dom}(\mathcal{A}))^n \mid (\mathcal{A}, \alpha[X_i \mapsto M_i]_{i=1}^n) \models \exists Y_1.\cdots\exists Y_m.\psi_1\}\big)$$

Moreover, we apply the correspondence between structures over Γ and partitions of the domain that we established before. We continue

$$= \left(\bigotimes_{i=1}^{n} \mathrm{B}_{p_i}\right)\big(\{(M_1,\dots,M_n) \in \mathcal{P}(\mathrm{dom}(\mathcal{A}))^n \mid \exists \mathcal{A}' \colon h(\mathcal{A}') = \mathcal{A},\ \mathcal{A}' \models \psi,\ (\mathrm{label}_a^{\mathcal{A}'})_{a\in\Gamma} \text{ is a partition of } \mathrm{dom}(\mathcal{A}'),\ f(\mathbb{1}_{M_1}(x),\dots,\mathbb{1}_{M_n}(x)) = g(a) \text{ for all } a \in \Gamma, x \in \mathrm{label}_a^{\mathcal{A}'}\}\big)$$

Taking the preimages under the mappings $(\{0,1\}^n)^{\mathrm{dom}(\mathcal{A})} \to (\{0,1\}^{\mathrm{dom}(\mathcal{A})})^n$ and $(\{0,1\}^{\mathrm{dom}(\mathcal{A})})^n \to (\mathcal{P}(\mathrm{dom}(\mathcal{A}))^n$, which are both measurable, we obtain

$$= \left(\bigotimes_{i=1}^{n} \mathrm{B}_{p_i}\right)\big(\{u \in (\{0,1\}^n)^{\mathrm{dom}(\mathcal{A})} \mid \exists \mathcal{A}' \colon (\mathcal{A}') = \mathcal{A},\ \mathcal{A}' \models \psi\ (\mathrm{label}_a^{\mathcal{A}'})_{a\in\Gamma} \text{ is a partition of } \mathrm{dom}(\mathcal{A}'),\ f(u(x)) = g(a) \text{ for all } a \in \Gamma, x \in \mathrm{label}_a^{\mathcal{A}'}\}\big)$$

By application of Proposition 5.21 we get

$$= \mathrm{B}_d\big(\{u \in M^{\mathrm{dom}(\mathcal{A})} \mid \exists \mathcal{A}' \colon h(\mathcal{A}') = \mathcal{A},\ \mathcal{A}' \models \psi\ (\mathrm{label}_a^{\mathcal{A}'})_{a\in\Gamma} \text{ is a partition of } \mathrm{dom}(\mathcal{A}')\ u(x) = g(a) \text{ for all } a \in \Gamma, x \in \mathrm{label}_a^{\mathcal{A}'}\}\big). \tag{5.6}$$

w. Assume $\mathcal{A} \in C$ is a $\mathcal{W}_\Sigma$-structure, i.e., $\mathcal{A} = \widetilde{w}$ for some $w \in \Sigma^\infty$. A $\mathcal{W}_\Gamma$ structure $\mathcal{A}'$ with label relations partitioning the domain and $h(\mathcal{A}') = \mathcal{A}$ corresponds to the word $w' = (w'_i)_{i\in\mathrm{pos}(w)} \in \Gamma^\infty$ given by $w'_i = a$ iff $i \in \mathrm{label}_a^{\mathcal{A}'}$ for $a \in \Gamma$ and $i \in \mathrm{pos}(w)$. From $h(\mathcal{A}') = \mathcal{A}$ we conclude that $i \in \mathrm{label}_a^{\mathcal{A}'}$ implies $i \in \mathrm{label}_{h(a)}^{\mathcal{A}}$ for all $i \in \mathrm{pos}(w)$. Thus, $h(w') = w$. Moreover, for every word $w' \in \Gamma^\infty$ with $h(w') = w$, we have $h(\widetilde{w'}) = \widetilde{w}$. Furthermore, a function $u \colon \mathrm{pos}(w) \to M$ with $u(x) = g(a)$ for all $a \in \Gamma$ and $x \in \mathrm{label}_a^{w'}$ corresponds to a word in $u' \in M^\infty$ with $|u'| = |w'|$ and $g(w') = u'$. Thus, we can rewrite (5.6) as

$$\begin{aligned}(5.6) &= \mathrm{B}_d(\{u \in M^\infty \mid \exists w' \in L \colon g(w') = u \wedge h(w') = w\}) \\ &= (\mathrm{B}_d \circ g)(h^{-1}(\{w\}) \cap L)\end{aligned}$$

$$= S_+(w).$$

t. Let $\mathcal{A} \in C$ be a $\mathcal{T}_\Sigma$ structure, i.e., $\mathcal{A} = \widetilde{t}$ for some $t \in \mathrm{T}_\Sigma$. For a $\mathcal{T}_\Gamma$-structure $\mathcal{A}'$ with label relations partitioning the domain and $h(\mathcal{A}') = \mathcal{A}$, let $t' \in \mathrm{T}_\Gamma$ be the tree given by $\mathrm{pos}(t') = \mathrm{pos}(t)$ and $t'(x) = f$ iff $x \in \mathrm{label}_f^{\mathcal{A}'}$. As in the word case, we have that $t'(x) = f$ implies $t(x) = h(f)$. Since h is a relabelling, f and $h(f)$ have the same arity. Therefore, t' is well-defined as a tree and $h(t') = t$. Moreover, for every tree $t' \in \mathrm{T}_\Gamma$ with $h(t') = t$, we have $h(\widetilde{t'}) = \widetilde{t}$. We conclude

$$\begin{aligned}(5.6) &= \mathrm{B}_d(\{u \in M^{\mathrm{pos}(t)} \mid \exists t' \in L \colon g(t') = u \wedge h(t') = t\}) \\ &= (\mathrm{B}_d \circ g)(h^{-1}(\{t\}) \cap L) \\ &= S_+(t).\end{aligned}$$

Therefore, $[\![\varphi_1]\!] = S_+$. The only thing left to do is to fix the value for the empty structure. We define the probabilistic MSO formula φ by

$$\varphi = \big(\varphi_1 \wedge (\exists x.x = x)\big) \vee \big(S(\varepsilon) \wedge (\forall x.x \neq x)\big).$$

Clearly, $[\![\varphi]\!] = S$. This concludes the proof. ∎

Using Theorems 5.22 and 5.23 and Theorems 3.9 and 3.20 from Chapter 3, we immediately obtain the following two corollaries.

Corollary 5.24. Let Σ be a finite alphabet and $S \colon \Sigma^\infty \to [0,1]$ be any function. The following statements are equivalent:

1. $S = [\![\varphi]\!]$ for a probabilistic MSO sentence $\varphi \in \mathrm{PMSO}(\mathcal{W}_\Sigma)$,
2. $S = \|A\|$ for a probabilistic Muller-automaton A.

The translations are effective in both directions.

Corollary 5.25. Let Σ be a finite ranked alphabet and $S \colon \mathrm{T}_\Sigma \to [0,1]$ be any function. The following statements are equivalent:

1. $S = [\![\varphi]\!]$ for a probabilistic MSO sentence $\varphi \in \mathrm{PMSO}(\mathcal{T}_\Sigma)$,
2. $S = \|A\|$ for a bottom-up probabilistic tree automaton A.

The translations are effective in both directions.

Remark 5.26. The proofs of Corollaries 5.24 and 5.25 make a detour through Nivat representations for both directions. We are not aware of any direct proof showing the equivalence of probabilistic automata and probabilistic MSO logic.

Part II

Probabilistic Regular Expressions

Chapter 6

Probabilistic Regular Expressions on Words

Regular expressions were introduced by Kleene [K56] in the 1950s. Only some years later, regular expressions have been extended to the weighted setting by Schützenberger [S61]. Both models only consider finite words. Nowadays, regular expressions have spread through all of theoretical computer science and enjoy manifold applications and generalisations to many different settings.

In this chapter, we recall the definition of classical regular expressions in Section 6.1. Afterwards, we transfer the classical operations used in regular expressions to the probabilistic setting and also to infinite words in Section 6.2. Using these definitions, we introduce probabilistic regular expressions on finite and infinite words in Section 6.3 and give some basic properties. The last two sections contain the proof of the expressive equivalence of probabilistic regular expressions and probabilistic Muller-automata.

The results of this chapter have been published in [W14].

For the rest of this chapter, we fix a finite alphabet Σ.

6.1 Classical Regular Expressions

In this section, we will first recall Kleene's notion of classical regular expressions and afterwards state Schützenberger's extension to the weighted setting.

Rational or regular expressions are built from the empty set and single letters using the operations union, language concatenation and Kleene-iteration. Every well-formed term using these operations is a rational expression.

Definition 6.1. The set RE of all *regular expressions* or *rational expressions* is given in BNF by

$$E ::= \emptyset \mid a \mid E \cup E \mid E \cdot E \mid E^*,$$

where a ranges over all letters $a \in \Sigma$.

With each regular expression E we associate its language $\mathrm{L}(E)$. The definition of $\mathrm{L}(E)$ is given inductively on the structure of E below:

$$\begin{aligned} \mathrm{L}(a) &= \{a\}, & \mathrm{L}(E \cup F) &= \mathrm{L}(E) \cup \mathrm{L}(F), \\ \mathrm{L}(E^*) &= \mathrm{L}(E)^*, & \mathrm{L}(E \cdot F) &= \mathrm{L}(E) \cdot \mathrm{L}(F), \\ \mathrm{L}(\emptyset) &= \emptyset. \end{aligned}$$

We call any language $L \subseteq \Sigma^*$ *regular* or *rational*, if there is a regular expression E such that $\mathrm{L}(E) = L$.

The following theorem is a fundamental result in the theory of formal languages and became known as *Kleene's theorem* [K56].

Theorem 6.2. Let $L \subseteq \Sigma^*$ be any language. The following statements are equivalent:

1. $L = \mathrm{L}(A)$ for some finite automaton A.
2. $L = \mathrm{L}(E)$ for some regular expression E.

Regular expressions describe languages of finite words. There is a simple generalisation to cover languages containing finite and infinite words.

Definition 6.3. The set of all *ω-regular expressions* is given in BNF by

$$E ::= R \mid R \cdot E \mid E \cup E \mid R^\omega,$$

where R is any regular expression as defined in Definition 6.1.

The language $\mathrm{L}(E) \subseteq \Sigma^\infty$ defined by an expression E is defined by induction on the structure of E:

$$\begin{aligned} \mathrm{L}(R) &= \mathrm{L}_{\text{Def. 6.1}}(R) & \mathrm{L}(R \cdot E) &= \mathrm{L}(R) \cdot \mathrm{L}(E) \\ \mathrm{L}(E_1 \cup E_2) &= \mathrm{L}(E_1) \cup \mathrm{L}(E_2) & \mathrm{L}(E^\omega) &= \mathrm{L}(E)^\omega. \end{aligned}$$

A language $L \subseteq \Sigma^\infty$ is called *ω-regular* if there is an ω-regular expression E with $\mathrm{L}(E) = L$.

Using the distributivity of $\cdot$ over $\cup$, one shows that every ω-regular expression is equivalent to an ω-regular expression of the form $E_0 \cup \bigcup_{i=1}^{n} E_i F_i^\omega$, where the E_i and F_j are regular expressions.

Example 6.4. We now come back to the automaton A from Example 2.2. Recall that the language of this automaton is

$$\begin{aligned}\mathrm{L}(A) =& \{ \mathrm{a}u\mathrm{a} \in \Sigma^* \mid u \in \Sigma^*, \text{ every maximal sequence of b's in } u \text{ has even length} \} \\ & \cup \{ \mathrm{a}w \in \Sigma^\omega \mid |w|_b \text{ is infinite} \},\end{aligned}$$

where $\Sigma = \{\mathrm{a}, \mathrm{b}\}$. We now give an ω-regular expression E with $\mathrm{L}(E) = \mathrm{L}(A)$:

$$E = \mathrm{a}\big(\mathrm{a} \cup \mathrm{bb}\big)^* \mathrm{a} \cup \mathrm{a}\big(\mathrm{a}^*\mathrm{b}\big)^\omega.$$

For finite words, b occurs only in pairs. Thus, the number of consecutive b's is always even. For infinite words, we infinitely often concatenate words that end with b. Thus, obtaining an infinite word with infinitely many b's. Conversely, every finite word with an even number of b's in any consecutive sequence of b's can be decomposed into a sequence of a and bb. Furthermore, every infinite word containing an infinite number of b's can be decomposed into words of the form $\mathrm{a}^*\mathrm{b}$. This shows $\mathrm{L}(E) = \mathrm{L}(A)$.

Using Kleene's theorem for finite words, one also obtains the expressive equivalence of Büchi-automata and ω-regular expressions.

Theorem 6.5. Let $L \subseteq \Sigma^\omega$. The following statements are equivalent:

1. $L = \mathrm{L}(A)$ for a Büchi-automaton A.
2. $L = \mathrm{L}(E)$ for an ω-regular expression E.

6.2 Probabilistic Rational Operations

Before we define the syntax and semantics of probabilistic regular expressions, we introduce probabilistic versions of the rational operations. The definitions of sum, concatenation, and Kleene-star correspond to their counterparts in weighted regular expressions [S61] over the semiring $(\overline{\mathbb{R}_+}, +, \cdot, 0, 1)$, where $\overline{\mathbb{R}}_+ = \mathbb{R}_+ \cup \{\infty\}$. The sum and product is extended to ∞ by letting $s + \infty = \infty$ for all $s \in \overline{\mathbb{R}}_+$, $s \cdot \infty = \infty$ for all $s > 0$ and $0 \cdot \infty = 0$.

For the rest of this chapter, $\mathbb{1}$ denotes the function with $\mathbb{1}(w) = 1$ for all $w \in \Sigma^\infty$.

Formally, given two functions $f, g \colon \Sigma^* \to \overline{\mathbb{R}}_+$, the operations *weighted concatenation* and *weighted Kleene-star* are defined as follows:

$$(f \cdot g)(w) = \sum_{uv=w} f(u)\, g(v), \qquad (f^*)(w) = \sum_{n \geq 0} f^n(w),$$

where $f^0 = \mathbb{1}_{\{\varepsilon\}}$ and $f^{n+1} = f \cdot f^n$. Then, $f \cdot g$ and f^* are again functions $\Sigma^* \to \overline{\mathbb{R}}_+$. Remark that in the semiring $\overline{\mathbb{R}}_+$ every countable sum converges to a value of the semiring. Thus, the Kleene-star of a function is always defined. Another common approach is to require $f(\varepsilon) = 0$. This is not needed here.

We would like to use these definitions also in our probabilistic setting. Unfortunately, even when f and g only attain values from the interval $[0, 1]$, the values of $f \cdot g$ and f^* may be unbounded. Therefore, we first give sufficient conditions such that the values of concatenation and Kleene-iteration are again interpretable as probability values.

Definition 6.6. Let $f \colon \Sigma^\infty \to [0, 1]$. We call f *prefix summable* if

$$\sum_{u \leq w} f(u) \leq 1$$

for all $w \in \Sigma^*$.

This definition is a generalisation of prefix free languages: a language $L \subseteq \Sigma^\infty$ is prefix free if it does not contain two words u and v with $u \prec v$, cf. Definition 6.29. Thus, a language L is prefix free if and only if $\mathbb{1}_L$ is prefix summable.

If a language L is prefix free and $w \in L\Sigma^*$, then there are unique words u, v with $u \in L$ such that $uv = w$. This property transfers to prefix summable series, which allows us to interpret the values of weighted concatenation as probability values.

Lemma 6.7. Let $f, g \colon \Sigma^\infty \to [0, 1]$ such that f is prefix summable. The series $f \cdot g$, defined by

$$(f \cdot g)(w) = \sum_{uv=w} f(u)\, g(v),$$

is bounded by 1. Moreover, if g is also prefix summable, so is $f \cdot g$.

Proof. Let $w \in \Sigma^\infty$. We obtain $(f \cdot g)(w) = \sum_{uv=w} f(u)\, g(v) \leq \sum_{uv=w} f(u) \leq 1$, as f is prefix summable. Thus, $f \cdot g$ is well-defined. Now, assume that g is also prefix summable. We compute

$$\sum_{uv=w} (f \cdot g)(u) = \sum_{uv=w} \sum_{xy=u} f(x)\, g(y) = \sum_{xv'=w} f(x) \sum_{v'=yv} g(y) \leq 1,$$

since g and f are prefix summable. Thus, $f \cdot g$ is prefix summable. ∎

In order to obtain a probabilistic ω-iteration, note that for a prefix free language $L \subseteq \Sigma^*$, the equation $L^\omega = \bigcap_{n \geq 1} L^n \Sigma^\omega$ holds, see Lemma 6.31. Hence, L^ω can be regarded as the limit of the sequence $L^n \Sigma^\omega$. By transferring this idea to the probabilistic setting, we obtain the probabilistic ω-iteration.

Definition 6.8. Given a prefix summable series $f\colon \Sigma^\infty \to [0,1]$, we define the *probabilistic ω-iteration* f^ω by

$$f^\omega(w) = \lim_{n\to\infty} (f^n \cdot \mathbb{1})(w),$$

where $\mathbb{1}\colon \Sigma^\infty \to [0,1]$ is the constant function with $\mathbb{1}(w) = 1$.

Lemma 6.9. Let f be a prefix summable series. Then, the series f^ω is a well-defined function $\Sigma^\infty \to [0,1]$.

Proof. By Lemma 6.7, we know that $(f^n \cdot \mathbb{1})(w) \leq 1$ for all $n \geq 0$ and $w \in \Sigma^\infty$. We show $(f^{n+1} \cdot \mathbb{1})(w) \leq (f^n \cdot \mathbb{1})(w)$: let $w \in \Sigma^\infty$.

$$\begin{aligned}(f^{n+1} \cdot \mathbb{1})(w) &= \sum_{u_1\cdots u_{n+1}v = w} \prod_{i=1}^{n+1} f(u_i) = \sum_{u_1\cdots u_n v' = w} \prod_{i=1}^{n} f(u_i) \sum_{u_{n+1}v = v'} f(u_{n+1}) \\ &\leq \sum_{u_1\cdots u_n v' = w} \prod_{i=1}^{n} f(u_i) \;= (f^n \cdot \mathbb{1})(w)\end{aligned}$$

Thus, the sequence $\big((f^n \cdot \mathbb{1})(w)\big)_{n\geq 0}$ is monotonically decreasing and bounded by 0. Therefore, the sequence converges with limit between 0 and $(f^0 \cdot \mathbb{1})(w) = 1$. ∎

Finally, we consider the Kleene-iteration. Intuitively, every step of Kleene-iteration involves two choices: whether to continue the iteration at all and if so, which word to choose. The choice of the next word is well-behaved for prefix summable series. To handle the exit condition, we require an extra series g, which is appended after f^*.

Definition 6.10. Let $f, g\colon \Sigma^\infty \to [0,1]$. We call the pair (f,g) an *iteration pair* if and only if

$$\sum_{u\leq w} f(u) + g(w) \leq 1,$$

for all $w \in \Sigma^\infty$.

Lemma 6.11. Let $f\colon \Sigma^\infty \to [0,1]$ be a prefix summable function. Furthermore, let $g\colon \Sigma^\infty \to [0,1]$ such that (f,g) is an iteration pair. Then, the series $(f^* \cdot g) + f^\omega$ is bounded by 1. Moreover, if $f + g$ is prefix-summable, so is $f^* \cdot g$.

Proof. Let $\mathbb{1}$ be the constant 1 function. For two functions $f_1, f_2\colon \Sigma^\infty \to \overline{R}_+$ let $f_1 \leq f_2$ if $f_1(w) \leq f_2(w)$ for all $w \in \Sigma^\infty$. Note that the probabilistic concatenation is monotonic in both arguments. Let g be a function such that (f,g) is an iteration

pair. We show $f^{k+1} \cdot \mathbb{1} + \sum_{n=0}^{k} f^n \cdot g \leq \mathbb{1}$ using induction on k. For $k = 0$ the statement is just the assumption that (f, g) is an iteration pair. We have for $k + 1$:

$$f^{k+2} \cdot \mathbb{1} + \sum_{n=0}^{k+1} f^n \cdot g = f^{k+1} \cdot (f \cdot \mathbb{1} + g) + \sum_{n=0}^{k} f^n \cdot g \leq f^{k+1} \cdot \mathbb{1} + \sum_{n=0}^{k} f^n \cdot g \leq \mathbb{1}.$$

Thus, the series $\sum_{n=0}^{k}(f^n \cdot g)(w)$ converges for $k \to \infty$ for every w since it is bounded and monotonically increasing. As the limit of $f^{k+1} \cdot \mathbb{1}$ is f^ω, we obtain $\sum_{n\geq 0}(f^n \cdot g)(w) + f^\omega(w) \leq 1$. Using the absolute convergence of $\sum_{n\geq 0}(f^n \cdot g)(w)$, we can rearrange this sum to obtain the desired bound for $f^* \cdot g$:

$$\sum_{n\geq 0}(f^n \cdot g)(w) = \sum_{n\geq 0}\sum_{uv=w} f^n(u)\, g(v) = \sum_{uv=w}\sum_{n\geq 0} f^n(u)\, g(v) = (f^* \cdot g)(w)$$

Assume that $f + g$ is prefix summable. Thus, $(f, g \cdot \mathbb{1})$ is an iteration pair and the function $f^* \cdot (g \cdot \mathbb{1})$ is bounded by 1. By the associativity of the weighted concatenation, we know that $(f^* \cdot g) \cdot \mathbb{1} = (f^* \cdot g) \cdot \mathbb{1}$. Hence, $f^* \cdot g$ is prefix summable. ∎

6.3 Probabilistic Regular Expressions

We introduce the syntax and semantics of probabilistic regular expressions. Furthermore, we state some basic semantic equalities.

As seen in the last section, the usual approach to define regular expressions on infinite words, is to first define expressions on finite words, and extend these to infinite words in a second step. For the probabilistic setting, we have to ensure that whenever a function $f^* \cdot g$ occurs in the semantics of an expression (f, g) is an iteration pair. Thus, we cannot use such a two parted definition, but have to define expressions on finite and infinite words simultaneously.

In the following definition we use a new symbol $\square$, which serves as a placeholder in regular expressions for places where other regular expressions can be inserted. This is necessary as we can only append to expressions which generate prefix summable series.

Definition 6.12. The set *PRE* all *probabilistic regular expressions* is the smallest set R which satisfies the following conditions:

1. $\square \in R$
2. If $A \subseteq \Sigma$ and $E_a \in R$ for $a \in A$, then $\sum_{a\in A} aE_a \in R$, and $\varepsilon + \sum_{a\in A} E_a \in R$

3. If $p \in [0,1]$, $E \in R$, and $F \in R$, then $pE + (1-p)F \in R$ and $pE \in R$

4. If $E\Box \in R$ and $F \in R$, then $EF \in R$

5. If $E\Box + F \in R$, then $E^*F + E^\omega \in R$, $E^*F \in R$, and $E^\omega \in R$,

and is closed under the following identities modelling the usual associativity, commutativity, and distributivity laws:

6. $E + (F + G) \equiv (E + F) + G$ and $E \cdot (F \cdot G) \equiv (E \cdot F) \cdot G$

7. $E + F \equiv F + E$

8. $E \cdot (F + G) \equiv EF + EG$ and $(E + F) \cdot G \equiv EG + FG$

Each identity states that an expression containing the left side of an identity as a subexpression is in R if and only if the same expression, but with this subexpression replaced by the right side of the identity, is in R and vice versa.

As in [BGMZ12], we call the rules 6 to 8 ACD rules.

We say an expression $E \in PRE$ is a *partial expression* if $\Box$ occurs within E, otherwise we say that E is *complete*.

Any subterm of an expression is called a *subexpression*. Note that a subexpression may not to be an expression.

Note that, the symbol $\Box$ only occurs in the syntax of probabilistic regular expressions, but not as actual symbol in the alphabet. Its entire use is to give a concise grammar for PRE. This is different from the use of variables in regular tree expressions which actually do occur as distinct letters in the ranked alphabet.

Next, we give the semantics of a PRE as function mapping finite or infinite words to probability values. Even though the symbol $\Box$ is only used as a placeholder expression in the syntax of PRE, we choose to give a meaningful semantics to the symbol. This will simplify further definitions.

The definition below states the semantics of a probabilistic regular expression as function mapping to $\overline{\mathbb{R}}_+$ and not to $[0,1]$. We will see afterwards that by the choice of the syntax of probabilistic regular expressions, any valid expression's semantics actually only attains values in $[0,1]$. Nevertheless, subexpressions may violate this property. Consider for example the expression $E = (1/2\ \mathrm{a} + 1/2\varepsilon)^*\ 1/2\ \varepsilon$: using the definition below, one can see that $\|E\|(a^n) = 1$ for all $n \geq 0$. Nevertheless, $\|(1/2\ \mathrm{a} + 1/2\varepsilon)^*\|(a^n) = 2$ for all $n \geq 0$.

The following definition gives the semantics of a PRE using structural induction on the syntax tree.

Definition 6.13. Let E be a PRE and $w \in \Sigma^\infty$. The semantics of E is a mapping $\|E\| \colon \Sigma^\infty \to \overline{\mathbb{R}}_+$ inductively defined by

$$\|a\|(w) = \begin{cases} 1 & \text{if } w = a \\ 0 & \text{otherwise,} \end{cases} \qquad \|p\|(w) = \begin{cases} p & \text{if } w = \varepsilon \\ 0 & \text{otherwise,} \end{cases}$$
$$\|E + F\|(w) = (\|E\| + \|F\|)(w), \qquad \|E^*\|(w) = (\|E\|^*)(w),$$
$$\|E \cdot F\|(w) = (\|E\| \cdot \|F\|)(w), \qquad \|E^\omega\|(w) = (\|E\|^\omega)(w),$$
$$\|\Box\|(w) = 1,$$

for all $w \in \Sigma^\infty$, $a \in \Sigma \cup \{\varepsilon\}$, and $p \in [0,1]$.

We show that the semantics of a probabilistic regular expression is always defined and attains a value in $[0,1]$. Before we prove this statement, we introduce the terms of an expression, which are independent of the application of the ACD rules. In the following definition the notation $\{\!\{\ldots\}\!\}$ is used to describe a multi-set, i.e., a set with multiplicities. The terms of an expression already appeared in [BGMZ12]. We extend their definition by splitting the set of terms in head terms and tail terms.

Definition 6.14. Let E be a PRE. We define the set $\mathcal{T}(E)$ of all *terms of* E inductively by

$$\mathcal{T}(x) = \{\!\{x\}\!\} \qquad \text{for } x \in A \cup \{\varepsilon, \Box\},$$
$$\mathcal{T}(E + F) = \mathcal{T}(E) \cup \mathcal{T}(F),$$
$$\mathcal{T}(E \cdot F) = \{\!\{ E' \cdot F' \;\big|\; E' \in \mathcal{T}(E),\ F' \in \mathcal{T}(F) \}\!\},$$
$$\mathcal{T}(E^*) = \{\!\{E^*\}\!\},$$
$$\mathcal{T}(E^\omega) = \{\!\{E^\omega\}\!\}.$$

Furthermore, we define the set $\mathrm{HT}(E)$ of all *head terms of* E and the set $\mathrm{TT}(E)$ of all *tail terms of* E by

$$\mathrm{HT}(E) = \{\!\{ E' \mid E'\Box \in \mathcal{T}(E) \}\!\},$$
$$\mathrm{TT}(E) = \{\!\{ E \in \mathcal{T}(E) \mid E \neq E'\Box \text{ for all } E' \in \mathrm{HT}(E) \}\!\}.$$

Intuitively, the terms of an expression are all summands that occur after applying distributivity until no product can be expanded. The head terms are all such summands whose last factor is $\Box$, and the tail terms are all other summands.

Since we would like the formula $\mathcal{T}(E) = \mathrm{HT}(E)\Box \cup \mathrm{TT}(E)$ to hold for all expressions E, we say $\mathrm{HT}(\Box)$ is just the empty expression (not to be mixed with the expression "ε"). In the next lemma we implicitly assume $\mathbb{1}_{\{\varepsilon\}}$ as the semantics of the empty expression.

Lemma 6.15. Let E be a probabilistic regular expression. Then, $\|E\|$ is a well-defined function $\Sigma^\infty \to [0,1]$.

Proof. Given an expression E, we need to show two conditions: that f^* and f^ω is only applied to prefix summable functions f, and that $\|E\| \leq 1$.

For a probabilistic regular expression E, let $\mathrm{H}(E) = \sum_{T \in \mathrm{HT}(E)} \|T\|$ and $\mathrm{T}(E) = \sum_{T \in \mathrm{TT}(E)} \|T\|$. Let the set M contain all probabilistic regular expressions E, such that the following conditions hold:

1. The operations * and $^\omega$ are only applied to expressions with prefix summable semantics in E, $\|E\| \leq 1$, and $\|E\| = \sum_{T \in \mathcal{T}(E)} \|T\|$
2. $(\mathrm{H}(E), \mathrm{T}(E))$ is an iteration pair

We prove that $M = \mathrm{PRE}$ by showing that M satisfies call conditions of Definition 6.12.

Clearly, $\square \in M$ holds. Let $A \subseteq \Sigma$ and assume expressions $E_a \in M$ for each $a \in A$. Let $E = \varepsilon + \sum_{a \in A} aE_a$. We show $E \in M$. There are no new expressions of the form F^* or F^ω in E, thus, by assumption on the E_a's, every iteration in E is only applied to a prefix summable function. Let $w \in \Sigma^\infty$. If $w = \varepsilon$, we have $\|E\|(w) = 1$. Otherwise, $w = aw'$, and we conclude $\|E\|(w) = 0$ if $a \notin A$ or $\|E\|(w) = \|E_a\|(w') \leq 1$ if $a \in A$, as $E_a \in M$. We show that $(\mathrm{H}(E), \mathrm{T}(E))$ is an iteration pair. By definition of E, $\mathrm{HT}(E) = \bigcup_{a \in A} a\,\mathrm{HT}(E_a)$ and $\mathrm{TT}(E) = \{\{\varepsilon\}\} \cup \bigcup_{a \in A} a\,\mathrm{TT}(E_a)$. Let $w \in \Sigma^\infty$. We show $\sum_{u \leq w} \mathrm{H}(E)(u) + \mathrm{T}(E)(w) \leq 1$. If $w = \varepsilon$ the statement follows directly from the definition. Assume $w = a_0 w'$. We obtain

$$\sum_{u \leq w} \mathrm{H}(E)(u) + \mathrm{T}(E)(w) = \sum_{u \leq w} \sum_{a \in A} \big(\|a\|\,\mathrm{H}(E_a)\big)(u) + \mathbb{1}_{\{\varepsilon\}}(w) + \sum_{a \in A} \big(\|a\|\,\mathrm{T}(E_a)\big)(w)$$

If $a_0 \notin A$, we immediately conclude that the value of this expression is 0. If $a_0 \in A$, only the summands that start with $\|a_0\|$ contribute a positive value. We continue

$$= \sum_{u' \leq w'} \mathrm{H}(E_{a_0})(u') + \mathrm{T}(E_{a_0})(w') \leq 1.$$

Therefore, $(\mathrm{H}(E), \mathrm{T}(E))$ is an iteration pair.

We consider the case $E = pE_1 + (1-p)E_2$. The proof of conditions 1. and 2. is analogous to the previous case and therefore left out here.

Assume $E_1\square, E_2 \in M$. As $(\mathrm{H}(E_1\square), \mathrm{T}(E_1\square))$ is an iteration pair by assumption and $\mathrm{HT}(E_1\square) = \mathcal{T}(E_1)$, we obtain that $\|E_1\|$ is prefix summable. Hence, $\|E_1\|\|E_2\|$ is well-defined by Lemma 6.7. Let $w \in \Sigma^\infty$, we conclude

$$\sum_{u \leq w} \mathrm{H}(E_1E_2)(u) + \mathrm{T}(E_1E_2)(w)$$

$$
\begin{aligned}
&= \sum_{u \leq w} \sum_{u_1 u_2 = u} \|E_1\|(u_1)\,\mathrm{H}(E_2)(u_2) + \sum_{uv=w} \|E_1\|(u)\,\mathrm{T}(E_2)(v) \\
&= \sum_{uv=w} \|E_1\|(u) \left(\sum_{v=v_1 v_2} \mathrm{H}(E_2)(v_1) + \mathrm{T}(E_2)(v) \right) \\
&\leq 1.
\end{aligned}
$$

Thus, $(\mathrm{H}(E), \mathrm{T}(E))$ is an iteration pair and $E \in M$.

We continue with the case $E = E_1 \Box + E_2 \in M$. We have $\mathrm{HT}(E) = \mathcal{T}(E_1) \cup \mathrm{HT}(E_2)$ and $\mathrm{TT}(E) = \mathrm{TT}(E_2)$. By assumption, $(\mathrm{H}(E), \mathrm{T}(E))$ is an iteration pair. In particular, $\|E_1\|$ is prefix summable. Hence, $\|E_1\|^*$ and $\|E_1\|^\omega$ are well-defined functions to $\mathbb{R}_+$. As $(\mathrm{H}(E_1), \mathrm{H}(E_2) \cdot \mathbb{1} + \mathrm{T}(E_2))$ is also an iteration pair, we additionally have $\|E_1\|^*(\mathrm{H}(E_2) \cdot \mathbb{1} + \mathrm{T}(E_2)) + \|E_1\|^\omega = \sum_{T \in \mathrm{HT}(E_2)} \|E_1^* T\| \cdot \mathbb{1} + \sum_{T \in \mathrm{TT}(E_2)} \|E_1^* E_2\| + E_1^\omega \leq 1$. Therefore, $(\mathrm{H}(E_1^* E_2 + E^\omega), \mathrm{T}(E_1^* E_2 + E^\omega))$ is an iteration pair. We conclude $E_1^* E_2 + E_1^\omega \in M$.

The set M is also closed under application of the ACD rules, as the terms of an expression do not change by application of these rules. ∎

Example 6.16. We return to the communication device introduced in Example 2.19. To build an expression for this model, we again consider the two letter alphabet $\Sigma = \{\mathsf{w}, \mathsf{i}\}$ for the events "wait" and "input message". We claim that the following expression models the probability that the buffer does not overflow:

$$
E = \Big(\mathsf{w} + \mathsf{i}\, p + \mathsf{i}\,(1-p) \big((1-q)\,\mathsf{w} + p\,\mathsf{i} \big)^* q\,\mathsf{w} \Big)^\omega .
$$

The intuition for E is as follows: the expression in the ω operator is the probability to return to the empty buffer state when starting with an empty buffer. If no new message is received, or a new input messages is received and it can be successfully sent right away, the buffer stays empty. In case of a new input message that fails to be sent successfully, this happens with probability $1 - p$, the device will try to send this message on every wait event. This fails with probability $1 - q$, and eventually succeeds with probability q. Any new incoming messages in this state must succeed to be sent immediately.

We still need to show that E is really a probabilistic regular expression, i.e., that it can be constructed using the rules given in Definition 6.12. First we show how to construct the expression $\big((1-q)\,\mathsf{w} + p\,\mathsf{i}\big)^* q\,\mathsf{w}\,\Box$.

$\Box \rightsquigarrow q\,\Box + (1-q)\,\Box$	Definition 6.12 (3)
$\rightsquigarrow \mathsf{w}\big(q\,\Box + (1-q)\,\Box\big) + \mathsf{i}\,p\,\Box$	Definition 6.12 (2), $p\Box$ was obtained using Definition 6.12 (3)

$$\begin{aligned}
&\rightsquigarrow \big((1-q)\,\mathsf{w} + p\,\mathsf{i}\big)\,\Box + q\,\mathsf{w}\Box && \text{Using ACD rules}\\
&\rightsquigarrow \big((1-q)\,\mathsf{w} + p\,\mathsf{i}\big)^* q\,\mathsf{w}\,\Box && \text{Definition 6.12 (5).}
\end{aligned} \tag{6.1}$$

We continue and construct the expression E:

$$\begin{aligned}
\Box &\rightsquigarrow p\,\Box + (1-p)\big((1-q)\,\mathsf{w} + p\,\mathsf{i}\big)^* q\,\mathsf{w}\,\Box && \text{Definition 6.12 (3) + (6.1)}\\
&\rightsquigarrow \mathsf{w}\,\Box + \mathsf{i}\,p\,\Box + \mathsf{i}\,(1-p)\big((1-q)\,\mathsf{w} + p\,\mathsf{i}\big)^* q\,\mathsf{w}\,\Box && \text{Definition 6.12 (2)}\\
&\rightsquigarrow \Big(\mathsf{w} + \mathsf{i}\,p + \mathsf{i}\,(1-p)\big((1-q)\,\mathsf{w} + p\,\mathsf{i}\big)^* q\,\mathsf{w}\Big)\,\Box && \text{Using ACD rules}\\
&\rightsquigarrow \Big(\mathsf{w} + \mathsf{i}\,p + \mathsf{i}\,(1-p)\big((1-q)\,\mathsf{w} + p\,\mathsf{i}\big)^* q\,\mathsf{w}\Big)^{\omega} && \text{Definition 6.12 (5)}
\end{aligned}$$

This shows that E is actually a probabilistic regular expression as defined in Definition 6.12.

Definition 6.17. Let E and F be two PREs. We say that E and F are *equivalent* if $\|E\|(w) = \|F\|(w)$ for all $w \in \Sigma^\infty$. In this case we write $E \equiv F$.

Next, we show two useful rules for building probabilistic regular expressions. The first rule states that any $\Box$ can be replaced by an arbitrary expression. The second rule allows us to omit summands from an expression.

Lemma 6.18. The following statements hold:

1. Let $E\Box + F$ and G be PRE, then $EG + F$ is also a PRE.
2. If $E + F$ is a PRE, so is E.

In order to prove Lemma 6.18 we need the following technical lemma.

Lemma 6.19. Let E be a probabilistic regular expression. The following statements hold:

1. $\sum_{T \in \mathcal{T}(E)} T \in \mathrm{PRE}$ and $\sum_{T \in \mathcal{T}(E)} T \equiv E$.
2. Let $M \subseteq \mathcal{T}(E)$ be a multi-set and $E_1, F_1, \ldots, E_n, F_n$ be subexpressions such that $E_i\Box \notin M$ for all $1 \leq i \leq n$ and $M \cup \{\!\{E_1\Box, \ldots, E_n\Box\}\!\} \subseteq \mathcal{T}(E)$. Then $\sum_{T \in M} T + \sum_{i=1}^{n} E_i F_i \in \mathrm{PRE}$. This is also true, if $M = \emptyset$ or $n = 0$.

Proof. The first statement can be shown by proving that the set of all expressions E which satisfy statement 1 satisfies the conditions given in Definition 6.12 and thus equals to PRE.

We show the second statement using the same technique. Let R be the set of all expressions satisfying statement 2. Clearly, the statement holds for $\Box$.

Let $A \subseteq \Sigma$ and $E_a \in R$ for all $a \in A$ and assume $E = \varepsilon + \sum_{a\in A} aE_a$. By definition, we have $\mathcal{T}(E) = \{\!\{\varepsilon\}\!\} \cup \bigcup_{a\in A}\{\!\{ aT \mid T \in \mathcal{T}(E_a)\}\!\}$. Let $M_a = \{\!\{T \mid aT \in M\}\!\}$ and $E_i = a_i E_i'$ with $a_i \in A$ and subexpressions E_i' for all $1 \le i \le n$. As $E_a \in R$, we have $\sum M_a + \sum_{i=1,\, a_i=a}^{n} E_i'F_i \in \mathrm{PRE}$. By Definition 6.12, we obtain

$$\varepsilon + \sum_{a\in A} a\left(\sum_{aT\in M} T + \sum_{i=1,\, a_i=a}^{n} E_i'F_i\right) \equiv \varepsilon + \sum_{aT\in M, a\in A} aT + \sum_{i=1}^{n}\sum_{a\in A, a=a_i} aE_i'F_i$$
$$\equiv \varepsilon + \sum_{T\in M} T + \sum_{i=1}^{n} E_iF_i \in \mathrm{PRE}.$$

The case $E = pF + (1-p)G$ is analogous to the previous case.

Let $E = FG$ with $F\Box, G \in R$. We have $\mathcal{T}(E) = \{\!\{T_1T_2 \mid T_1 \in \mathcal{T}(F),\, T_2 \in \mathcal{T}(G)\}\!\}$. Define multisets $M_T = \{\!\{T' \mid TT' \in M\}\!\}$ for every $T \in \mathcal{T}(F)$. Furthermore, let $E_i = T_iE_i'$ with $T_i \in \mathcal{T}(F)$ and $E_i' \in \mathcal{T}(G)$. As $G \in R$ it follows that $E_T = \sum M_T + \sum_{i=1,\, T_i=T}^{n} E_i'F_i \in \mathrm{PRE}$ for every $T \in \mathcal{T}(F)$. Next, we apply the hypothesis to $F\Box$ with $M = \emptyset$ and $\{\!\{T_1\Box, \ldots, T_n\Box\}\!\} = \mathcal{T}(F\Box)$ and $F_i = E_{T_i}$. As before, using distributivity, we obtain $E \in R$.

Finally, we consider the case $E = F^*G + F^\omega$ with $F\Box + G \in R$. All subexpressions E_i must be of the form $E_i = F^*E_i'$ with $E_i'\Box \in \mathcal{T}(G)$. Moreover, $M = \{\!\{F^*T \mid T \in M'\}\!\} \cup \{\!\{F^\omega \mid F^\omega \in M\}\!\}$ for some multiset $M' \subseteq \mathcal{T}(G)$. We apply the induction hypothesis to $F\Box + G$ with $M' \cup \mathcal{T}(F\Box)$ and subexpressions $E_1', F_1, \ldots, E_n', F_n$. Thus, $\sum_{T\in\mathcal{T}(F\Box)} T + \sum_{T\in\mathcal{T}(M')} T + \sum_{i=1}^{n} E_i'F_i$ is a probabilistic regular expression. As the first sum is equivalent to $F\Box$, we conclude the desired result by Definition 6.12 (5). ∎

The statements of Lemma 6.18 follow now directly from Lemma 6.19.

Proof (of Lemma 6.18). **1.** We apply Lemma 6.19 with $M = \mathcal{T}(F)$, $E_1 = E$, and $F_1 = G$. Thus, $M \cup \{\!\{E_1\Box\}\!\} = \mathcal{T}(E\Box + F)$ and we obtain $\sum_{T\in\mathcal{T}(F)} T + EG \equiv F + EG \in \mathrm{PRE}$.

2. Again, we use Lemma 6.19. This time, with $M = \mathcal{T}(E)$ and $n = 0$. We obtain $E = \sum_{T\in M} T \in \mathrm{PRE}$. ∎

6.4 From Expressions to Automata

In this section, we give a constructive proof that every probabilistic regular expressions admits an equivalent probabilistic Muller-automaton. The constructions are based on the ideas of [BGMZ12], but extended to the infinite word setting. Whereas the constructions themselves are not much more complicated than the constructions in the finite word case, showing their correctness on infinite words adds technical difficulties to the proofs.

For a probabilistic Muller-automaton $A = (Q, \delta, \mu, F, \mathcal{R})$ and a set $X \subseteq Q$, we denote the probabilistic Muller-automaton $(Q, \delta, \mu, X, \emptyset)$ by $A[X]_{\mathrm{F}}$. For a subset $\mathcal{X} \subseteq \mathcal{P}(Q)$, we write $A[\mathcal{X}]_{\mathcal{R}}$ for the automaton $(Q, \delta, \mu, \emptyset, \mathcal{X})$.

Definition 6.20. Let E be a PRE and $A = (Q, \delta, \mu, F, \mathcal{R})$ a probabilistic Muller-automaton. We say that A is an *automaton for* E if there is a partition $F = F_0 \cup \bigcup_{H \in \mathrm{HT}(E)} F_H$ of F, such that

1. $\sum_{E' \in \mathrm{TT}(E)} \|E'\| = \|A_0\|$ where $A_0 = (Q, \delta, \mu, F_0, \mathcal{R})$
2. $\|H\| = \left\| A[F_H]_{\mathrm{F}} \right\|$ for all $H \in \mathrm{HT}(E)$
3. The states in F_H are sinks for every $H \in \mathrm{HT}(E)$

Note that if E is a complete expression, i.e., $\mathrm{HT}(E) = \emptyset$, and A is an automaton for E, then the semantics of E already equals the behaviour of A. Thus, our goal for this section is to show that the set of expressions E, such that there is an automaton for E, satisfies the closure properties of Definition 6.12.

Lemma 6.21. There is an automaton for $\square$.

Proof. We have $\mathrm{HT}(\square) = \{\{\varepsilon\}\}$ and $\mathrm{TT}(\square) = \emptyset$. Thus, the automaton A given by $A = (\{q_0\}, \delta, \mathbb{1}_{\{q_0\}}, \{q_0\}, \emptyset)$ with $\delta(q_0, a) = \mathbb{0}$ for all $a \in \Sigma$ is an automaton for $\square$. ∎

Lemma 6.22. Let $\Gamma \subseteq \Sigma$ and E_a be a PRE for every $a \in \Gamma$. Furthermore, assume there is an automaton for each expression E_a. Then there are automata for $\varepsilon + \sum_{a \in \Gamma} aE_a$ and for $\sum_{a \in \Gamma} aE_a$.

Proof. Assume $E = \varepsilon + \sum_{a \in \Gamma} aE_a$ and $A_a = (Q_a, \delta_a, \mu_a, F_a, \mathcal{R}_a)$ is an automaton for E_a for every $a \in \Gamma$ such that the sets Q_a are pairwise disjoint. We define the automaton $A = (Q, \delta, \mu, F, \mathcal{R})$ by

$$Q = \{q_0, q_f\} \cup \bigcup_{a \in \Gamma} Q_a, \qquad \mu(q) = \mathbb{1}_{\{q_0\}}(q),$$

$$F = \{q_0\} \cup \bigcup_{a \in \Gamma} F_a, \qquad \mathcal{R} = \bigcup_{a \in \Gamma} \mathcal{R}_a,$$

$$\delta(q, a)(q') = \begin{cases} \mu_a(q') & \text{if } q = q_0,\ a \in \Gamma, \text{ and } q' \in Q_a \\ 1 & \text{if } q = q_0,\ a \notin \Gamma, \text{ and } q' = q_f \\ \delta_b(q, a, q') & \text{if } q, q' \in Q_b \text{ for some } b \in \Sigma \\ 1 & \text{if } q = q' = q_f \\ 0 & \text{otherwise.} \end{cases}$$

By construction, we have $\|A\|(\varepsilon) = 1$ and $\|A\|(aw) = \|A_a\|(w)$ for all $a \in \Gamma$. For $w' = aw$ with $a \notin \Gamma$, we obtain $\|A\|(w') = 0$, as A enters q_f after reading a with probability 1, which is not final, but cannot be left again.

We still need to show that A is an automaton for E. By definition we have $\mathrm{HT}(E) = \{\!\{ aE' \mid a \in \Sigma,\, E' \in \mathrm{HT}(E_a) \}\!\}$ and $\mathrm{TT}(E) = \{\!\{\varepsilon\}\!\} \cup \{\!\{ aE' \mid A \in \Sigma,\, E' \in \mathrm{TT}(E) \}\!\}$. Let $a \in \Gamma$. As A_a is an automaton for E_a there is a partition $F_a = F_a^0 \cup \bigcup_{E' \in \mathrm{HT}(E_a)} F_a^{E'}$ of F_a as in Definition 6.20. Let $aE' \in \mathrm{HT}(E)$ and $A' = A[F_a^{E'}]_{\mathrm{F}}$. By definition of A we have $\|A'\|(\varepsilon) = 0$ and $\|A'\|(bu) = \|A_a\|(u)$ if $b = a$ and $\|A'\|(bu) = 0$ otherwise. Thus, $\|A'\| = \|aE'\|$. On the other hand, let $A'' = (Q, \delta, \mu, \{q_0\} \cup \bigcup_{a \in \Sigma} F_a^0, \mathcal{R})$. Again by definition of A we conclude $\|A''\|(\varepsilon) = 1$ and $\|A''\|(aw) = \|A_a''\|(w)$, where $A_a'' = (Q_a, \delta_a, \mu_a, F_a^0, \mathcal{R}_a)$. By assumption on A_a, we have $\|A_a''\| = \sum_{E' \in \mathrm{TT}(E_a)} \|E'\|$ and so $\|A''\| = norm\varepsilon + \sum_{a \in \Sigma} \sum_{E'_a \in \mathrm{TT}(E_a)} \|aE'_a\| = \sum_{E' \in \mathrm{TT}(E)} \|E'\|$. Therefore, A is an automaton for E.

The case $E = \sum_{a \in A} aE_a$ is analogous, the only difference is to omit q_0 from F in the construction of A. ∎

Lemma 6.23. Let E and F be PREs which each admit an automaton. Furthermore, let $p \in [0,1]$. There is an automaton for $pE + (1-p)F$.

Proof. Let $A_i = (Q_i, \delta_i, \mu_i, F_i, \mathcal{R}_i)$ for $i = 1, 2$ such that A_1 is an automaton for E and A_2 is an automaton for F. We assume that Q_1 and Q_2 are disjoint. We define an automaton A by $A = (Q_1 \cup Q_2, \delta, \mu, F_1 \cup F_2, \mathcal{R}_1 \cup \mathcal{R}_2)$ and

$$\delta(p,a,q) = \begin{cases} \delta_i(p,a,q) & \text{if } p,q \in Q_i \text{ for } i = 1,2, \\ 0 & \text{otherwise,} \end{cases}$$

$$\mu(q) = \begin{cases} p\,\mu_1(q) & \text{if } q \in Q_1, \\ (1-p)\,\mu_2(q) & \text{if } q \in Q_2. \end{cases}$$

The automaton A chooses in its initial distribution a state from Q_1 with probability p and a state from Q_2 with probability $1-p$. Afterwards A simulates the automaton A_1 or A_2, respectively.

The proof that A is indeed an automaton for $pE + (1-p)F$ is left to the reader. ∎

Lemma 6.24. Let $E_1\Box$ and E_2 be expressions which both admit an automaton. There is an automaton for $E_1 \cdot E_2$.

Proof. Let $A_i = (Q_i, \delta_i, \mu_i, F_i, \mathcal{R}_i)$ for $i = 1, 2$ be probabilistic Muller-automata such that A_1 is an automaton for $E_1\Box$ and A_2 is an automaton for E_2. The new automaton A resembles the usual construction for the concatenation of regular languages: starting in A_1, transitions which might enter a final state in A_1 are detoured to the

initial states of A_2. Unfortunately, as we need the correspondence between final states of A and head terms of E_1E_2 we have to enlarge the state set to satisfy this condition.

Let $F_1 = F_1^0 \cup \bigcup_{G\in\mathrm{HT}(E_1\Box)} F_1^G$ and $F_2 = F_2^0 \cup \bigcup_{G\in\mathrm{HT}(E_2)} F_2^G$ as in Definition 6.20. Note that $\mathrm{TT}(E_1\Box) = \emptyset$. Thus, we may assume $F_1^0 = \emptyset$ and $\mathcal{R}_1 = \emptyset$. Formally, we define $A = (Q, \delta, \mu, F, \mathcal{R})$ by

$$
\begin{aligned}
Q &= Q_1 \setminus F_1 \cup (\mathrm{HT}(E\Box) \times Q_2)\\
\delta(p,a,q) &= \begin{cases} \delta_1(p,a,q) & \text{if } p,q \in Q_1 \setminus F_1\\ \delta_2(p',a,q') & \text{if } p = (G,p'),\ q = (G,q') \text{ for some } G \in \mathrm{HT}(E\Box)\\ \delta_1(p,a,F_1^G)\,\mu_2(q') & \text{if } p \in Q_1 \setminus F_1 \text{ and } q = (G,q') \text{ for some } G\\ 0 & \text{otherwise.}\end{cases}\\
\mu(q) &= \begin{cases} \mu_1(q) & \text{if } q \in Q_1 \setminus F_1\\ \mu_1(F_1^G)\,\mu_2(q') & \text{if } q = (G,q') \text{ for some } G \in \mathrm{HT}(E\Box)\end{cases}\\
F &= \big\{(G,q) \;\big|\; G \in \mathrm{HT}(E\Box),\ q \in F_2\big\}\\
\mathcal{R} &= \big\{\{(G,q) \mid q \in R\} \;\big|\; R \in \mathcal{R}_2,\ G \in \mathrm{HT}(E\Box)\big\}.
\end{aligned}
$$

Note that A is actually a probabilistic Muller automaton, i.e., $\delta(p,a)$ is a distribution for every possible choice of p and a.

We show that the constructed automaton is an automaton for EF. Before we prove the actual statement, we give the following auxiliary result: let $G \in \mathrm{HT}(E\Box)$, $Q_G = \{G\} \times Q_2$, and $\kappa_G \colon Q_2^\infty \to Q_G^\infty$ be the unique homomorphism with $\kappa_G(q) = (G,q)$. Then

$$
\Pr_A^w((Q_1 \setminus F_1)^n R) = \|A_1[F_1^G]_{\mathrm{F}}\|(w_1 \cdots w_n) \cdot \Pr_{A_2}^{w_{n+1}\cdots}(\kappa_G^{-1}(R)), \tag{6.2}
$$

for all measurable sets $R \subseteq Q_G Q_G^\infty$.

Let $R_q = \{\tau \in Q_G^\infty \mid (G,q)\,\tau \in R\}$, i.e., all words from R that start with (G,q) without the first letter. Then, $R = \bigcup_{q\in Q_2} (G,q)\,R_q$. Using Proposition 2.20 we conclude

$$
\begin{aligned}
\Pr_A^w((Q_1 \setminus F_1)^n R) &= \sum_{r_0,\cdots,r_{n-1}\in Q_1\setminus F_1} \sum_{q\in Q_2} \Pr_A^w(r_0 \cdots r_{n-1}(G,q)R_q)\\
&= \begin{cases} \displaystyle\sum_{r_0,\cdots,r_{n-1}\in Q_1\setminus F_1} \sum_{q\in Q_2} \mu_1(r_0)\left(\prod_{i=1}^{n-1} \delta(r_{i-1},w_i,r_i)\right)\delta(r_n,w_n,F_1^G) & \\ \qquad\qquad \cdot\,\mu_2(q)\,\Pr_{A_{(G,q)}}^{w_{n+2}\cdots}((G,q)R_q) & \text{if } n > 0\\ \displaystyle\sum_{q\in Q_2} \mu_1(F_1^G)\,\mu_2(q)\,\Pr_{A_{(G,q)}}^{w}((G,q)R_q) & \text{if } n = 0\end{cases}
\end{aligned}
$$

$$= \sum_{r_0,\dots,r_{n-1}\in Q_1\setminus F_1, r_n\in F_1^G} \mu_1(r_0)\prod_{i=1}^{n}\delta_1(r_{i-1},w_i,r_i)\Pr_{A_2}^{w_{n+1}\cdots}(\kappa_G^{-1}(R))$$
$$= \sum_{n\geq -1} \|A_1[F_1^G]_{\mathrm{F}}\|(w_1\cdots w_n)\cdot\Pr_{A_2}^{w_{n+1}\cdots}(\kappa_G^{-1}(R)).$$

This completes the proof of (6.2) and we are ready to prove the correctness of A.

For every $GH \in \mathrm{HT}(EF)$ with $G \in \mathrm{HT}(E\Box)$ and $H \in \mathrm{HT}(F)$ we define $F_{GH} = \{(G,q) \in Q \mid q \in F_2^H\}$. Moreover, we set $F_0 = \{(G,q) \mid G \in \mathrm{HT}(E\Box),\ q \in F_2^0\}$. Thus, $F_0 \cup \bigcup_{GH\in\mathrm{HT}(EF)} F_{GH}$ is a partition of F.

We show $\|A[F_{GH}]_{\mathrm{F}}\| = \|GH\|$ for all $GH \in \mathrm{HT}(EF)$. Let $w = w_1\cdots w_{|w|} \in \Sigma^*$. Note that, by the structure of automaton A, the set of runs with non-zero probability is contained in $\bigcup_{G\in\mathrm{HT}(E\Box)}(Q_1\setminus F_1)^*(\{G\}\times Q_2)^\infty$. Thus, taking the intersection of any measurable set M with this set does not change the probability of M. We compute

$$\begin{aligned}\|A[F_{GH}]_{\mathrm{F}}\|(w) &= \Pr_A^w(Q^{|w|}(\{G\}\times F_2^H))\\ &= \sum_{0\leq n\leq |w|}\Pr_A^w((Q_1\setminus F_1)^n Q_G^{|w|-n}(\{G\}\times F_2^H))\end{aligned}$$

Next, we apply (6.2):

$$\begin{aligned}&= \sum_{0\leq n\leq |w|}\|A_1[F_1^G]_{\mathrm{F}}\|(w_1\cdots w_n)\Pr_A^{w_{n+1}\cdots w_{|w|}}(\kappa_G^{-1}(Q_G^{|w|-n}(\{G\}\times F_2^H)))\\ &= \sum_{uv=w}\|A_1[F_1^G]_{\mathrm{F}}\|(u)\Pr_{A_2}^v(Q_2^{|v|}F_2^H)\end{aligned}$$

By our assumption on A_1 and A_2 we have $\|A_1[F_1^G]_{\mathrm{F}}\| = \|G\|$ and $\|A_2[F_2^H]_{\mathrm{F}}\| = \|H\|$. Thus

$$= (\|G\|\cdot\|H\|)(w) = \|GH\|(w).$$

Finally, we prove $\|A'\| = \sum_{T\in\mathrm{TT}(EF)}\|T\|$, where $A' = (Q,\delta,\mu,F_0,\mathcal{R})$. At first, we show the statement for finite words. Let $w = w_1\cdots w_{|w|} \in \Sigma^*$.

$$\|A[F_0]_{\mathrm{F}}\|(w) = \sum_{G\in\mathrm{HT}(E\Box)}\Pr_A^w(Q^{|w|}(\{G\}\times F_2^0))$$

As before, we split the runs in parts running in A_1 and in A_2:

$$= \sum_{G\in\mathrm{HT}(E\Box)}(\|A_1[F_1^G]_{\mathrm{F}}\|\cdot\|A_2[F_2^0]_{\mathrm{F}}\|)(w)$$

By assumption, we have $\|A_2[F_2^0]_{\mathrm{F}}\| = \sum_{H\in\mathrm{TT}(F)}\|H\|$:

$$= \sum_{G\in\mathrm{HT}(E\Box)}\sum_{H\in\mathrm{TT}(F)}(\|G\|\cdot\|H\|)(w)$$

$$= \sum_{T \in \mathrm{TT}(EF)} \|T\|(w).$$

For the infinite word case let $w \in \Sigma^\omega$. We obtain

$$\|A\|(w) = \Pr_A^w(\{\rho \in Q^\omega \mid \inf(\rho) \in \mathcal{R}\})$$

By construction $\mathcal{R}$ contains the κ_G images of the sets in $\mathcal{R}_2$:

$$= \sum_{G \in \mathrm{HT}(E\square)} \Pr_A^w(\{\rho \in Q^\omega \mid \kappa_G^{-1}(\inf(\rho)) \in \mathcal{R}_2\})$$

The complement of set $\bigcup_G (Q_1 \setminus F_1)^*(\{G\} \times Q_2)^\omega$ has probability zero:

$$= \sum_{G \in \mathrm{HT}(E\square)} \Pr_A^w((Q_1 \setminus F_1)^*\{\rho \in (\{G\} \times Q_2)^\omega \mid \kappa_G^{-1}(\inf(\rho)) \in \mathcal{R}_2\})$$

As before, we can seperate the runs in A_1 and A_2:

$$\begin{aligned}
&= \sum_{G \in \mathrm{HT}(E\square)} (\|A_1[F_1^G]_{\mathrm{F}}\| \cdot \underbrace{\Pr_{A_2}(\{\rho \in Q_2^\omega \mid \inf(\rho) \in \mathcal{R}_2\})}_{=\|A_2\|})(w) \\
&= \sum_{G \in \mathrm{HT}(E\square)} \sum_{H \in \mathrm{TT}(F)} (\|G\| \cdot \|H\|)(w) \\
&= \sum_{T \in \mathrm{TT}(EF)} \|T\|(w).
\end{aligned}$$

Therefore, A is an automaton for EF and the proof is complete. ∎

Our final step is to show that the recognizable series are also closed under iteration, i.e., rule Definition 6.12 (5). Before we can prove this result, we need a preparatory result which shows that the expected values the elements of a convergent sequence converge to the same value as the sequence itself.

We suppose that the next two results have already appeared in the literature on probability theory, but we could not find a concrete reference.

Lemma 6.25. Let $f : \mathbb{R} \to \mathbb{R}$ be a bounded, measurable function such that the limit $\lim_{x\to\infty} f(x)$ exists. Furthermore, let $(X_n)_{n\geq 1}$ be a sequence of random variables over probability spaces $(\Omega_i, \mathcal{A}_i, \Pr_i)$ such that

1. $|\mathbb{E}[X_n]| < \infty$ and $\sigma(X_n) < \infty$ for all $n \geq 1$,
2. $\mathbb{E}[X_n] \to \infty$ for $n \to \infty$,

3. $\sigma(X_n)/\mathbb{E}[X_n] \to 0$ for $n \to \infty$,

where the expected values and standard deviations are computed with respect to the corresponding probability spaces. Then, $\mathbb{E}[f(X_n)]$ converges for $n \to \infty$ and

$$\lim_{n\to\infty} \mathbb{E}[f(X_n)] = \lim_{x\to\infty} f(x).$$

Note that the requirement $\sigma(X_n)/\mathbb{E}[X] \to 0$ is really necessary. Consider the random variables X_n defined by $\Pr(X_n = 0) = 1/2$ and $\Pr(X_n = n) = 1/2$. Thus, $\mathbb{E}[X_n] = n/2$. On the other hand, let $f(x) = 1/\max(1,x)$. We obtain $\mathbb{E}[f(X_n)] = 1/2 + 1/2 \cdot 1/n \to 1/2 \neq 0 = \lim_{n\to\infty} f(n/2)$. This does not contradict the above lemma as $\sigma(X_n) = n/2$.

Proof (of Lemma 6.25). Let $a = \lim_{x\to\infty} f(x)$ and M be a bound of $|f|$. Let $\epsilon > 0$ be arbitrary. Choose a $C > 0$ such that $2M \leq \epsilon/2 C^2$ and N_0 large enough such that $|f(x) - a| \leq \epsilon/2$ for all $x \geq N_0$. Next, choose $N_1 \geq N_0$ such that $1/2\mathbb{E}[X_n] \geq N_0$ for all $n \geq N_1$. Finally choose $N_2 \geq N_1$ with $8M\sigma(X_n)^2/\mathbb{E}[X_n]^2 \leq \epsilon/2$ for all $n \geq N_2$.

We obtain for $n \geq N_2$:

$$\begin{aligned}
&|\mathbb{E}[f(X_n)] - a| \\
&\leq \int |f(X_n) - a| \, \mathrm{dPr} \\
&= \int_{|X_n - \mathbb{E}[X_n]| \geq 1/2\mathbb{E}[X_n]} |f(X_n) - a| \, \mathrm{dPr} + \int_{|X_n - \mathbb{E}[X_n]| < 1/2\mathbb{E}[X_n]} |f(X_n) - a| \, \mathrm{dPr}
\end{aligned}$$

As f is bounded by M and $|X_n - \mathbb{E}[X_n]| < 1/2\mathbb{E}[X_n]$ implies $X_n \geq 1/2\mathbb{E}[X_n] \geq N_0$ by the choice of N_1, we continue:

$$\leq 2M \Pr\left(|X_n - \mathbb{E}[X_n]| \geq \frac{1}{2}\mathbb{E}[X_n]\right) + \int_{X_n \geq N_0} |f(X_n) - a| \, \mathrm{dPr}$$

By Chebyshev's inequality and the choice of N_0:

$$\leq 2M \frac{\sigma(X_n)^2}{1/4\mathbb{E}(X_n)^2} + \frac{\epsilon}{2}$$

By the choice of N_2:

$$\leq \epsilon.$$

Hence, we obtain $\lim_{n\to\infty} \mathbb{E}[f(X_n)] = a$. ∎

Corollary 6.26. Let $(a_n)_{n\geq 0}$ a convergent sequence and $(X_n)_{n\geq 1}$ $\mathbb{N}_0$-valued random variables which satisfy conditions 1 to 3 of Lemma 6.25. Then

$$\lim_{n\to\infty} \sum_{k\geq 0} a_k \Pr(X_n = k) = \lim_{n\to\infty} a_n.$$

Proof. We define a function $f : \mathbb{R} \to \mathbb{R}$ by

$$f(x) = \begin{cases} a_n & \text{if } x \in [n, n+1) \text{ for some } n \in \mathbb{N}_0 \\ 0 & \text{otherwise.} \end{cases}$$

As $(a_n)_{n\geq 0}$ converges, so does f for $x \to \infty$ and the limits agree. Furthermore, also by the convergence of $(a_n)_{n\geq 0}$, f is bounded. Thus we can apply Lemma 6.25 and obtain

$$\lim_{n\to\infty} \sum_{k\geq 0} a_k \Pr(X_n = k) = \lim_{n\to\infty} \mathbb{E}[f(X_n)] = \lim_{x\to\infty} f(x) = \lim_{n\to\infty} a_n,$$

where the first equality holds as the X_n only attain values in $\mathbb{N}_0$. ∎

We are now ready to prove the closure of recognizable series under iteration.

Lemma 6.27. Let $E\Box + F$ be an expression which admits an automaton. There are automata for $E^*F + E^\omega$ and for E^ω and E^*F.

Proof. We show the "$E^*F + E^\omega$" case. The other cases are analogous. Let $A = (Q, \delta, \mu, X, \mathcal{R})$ be an automaton for $E\Box + F$.[1] There is a partition $X = X_0 \cup \bigcup_{E' \in \mathrm{HT}(E\Box+F)} X_{E'}$ such that Definition 6.20 holds. Let $X_E = \bigcup_{E' \in \mathrm{HT}(E\Box)} X_{E'}$. We may assume $\mu(X_E) < 1$. Otherwise, $\|E\| \equiv \mathbb{1}$ and $\|F\| \equiv \mathbb{0}$ as all states in X_E are sinks.

We construct an automaton A' which simulates the automaton A until it can reach a state from X_E. At this point, instead of entering X_E, the automaton accumulates the acceptance probability of the computation so far, and resets the simulated automaton to start a new computation. In order to define an acceptance condition based on the number of computations, we mark states that start a new computation. Moreover, whenever a new computation is started, we add a factor $\frac{1}{1-\mu(X_E)}$ to account for an arbitrary number of computations on the empty word.

For every $q \in Q$ let $\overline{q}$ be a new, marked state and for a set $P \subseteq Q$ let $\overline{P}$ contain all states $\overline{p}$ for $p \in P$. We write $\widetilde{q}$ if both q and $\overline{q}$ can be used, i.e., $r = \widetilde{q}$ stands for $r = q$ or $r = \overline{q}$. Define $A' = (Q', \delta', \mu', X', \mathcal{R}')$ by

$$Q' = Q_0 \cup \overline{Q_0} \text{ where } Q_0 = Q \setminus X_E,$$

$$\delta'(\widetilde{p}, a, \widetilde{q}) = \begin{cases} \delta(p, a, q) & \text{if } \widetilde{q} = q \\ \sum_{r \in X_E} \delta(p, a, r) \frac{\mu(q)}{1-\mu(X_E)} & \text{if } \widetilde{q} = \overline{q}, \end{cases}$$

[1] We use X for the final states (e**x**it states) to avoid the name clash

$$\mu'(\widetilde{q}) = \begin{cases} \frac{\mu(q)}{1-\mu(X_E)} & \text{if } \widetilde{q} = \overline{q} \\ 0 & \text{otherwise,} \end{cases}$$
$$X' = (X \setminus X_E) \cup \overline{X \setminus X_E},$$
$$\mathcal{R}' = \mathcal{R} \cup \mathcal{R}_1 \text{ where } \mathcal{R}_1 = \left\{ R \subseteq Q' \;\middle|\; \overline{Q_0} \cap R \neq \emptyset \right\}.$$

We show that A' is an automaton for $E^*F + E^\omega$. Note that $\mathrm{TT}(E^*F + E^\omega) = \{\!\{ E^*F' \mid F' \in \mathrm{TT}(F) \}\!\} \cup \{\!\{E^\omega\}\!\}$ and $\mathrm{HT}(E^*F + E^\omega) = \{\!\{ E^*F' \mid F' \in \mathrm{HT}(F) \}\!\}$.

Let $P \subseteq Q$ be a set of states, $R \subseteq Q'^\infty$ a measurable set of (finite or infinite) runs, and $n \geq 0$. Then for all $w \in \Sigma^\infty$:

$$\Pr^w_{A'}\left(\left(\overline{Q_0}Q_0^*\right)^n \overline{P}R\right) = \sum_{\substack{u_1 \cdots u_n v = w \\ u_1, \ldots, u_n \in \Sigma^+}} \left(\prod_{i=1}^{n} \frac{\|A[X_E]_{\mathrm{F}}\|(u_i)}{1 - \mu(X_E)} \right) \sum_{q \in P} \frac{\mu(q)}{1 - \mu(X_E)} \Pr^v_{A'_q}(qR). \quad (6.3)$$

Let $w = (w_i)_{i \in \mathrm{pos}(w)} \in \Sigma^\infty$. We compute

$$\Pr^w_A\left(\left(\overline{Q_0}Q_0^*\right)^n \overline{P}R\right)$$

Every run in $(\overline{Q_0}Q_0^*)^n \overline{P}R$ contains exactly $n+1$ positions from $\overline{Q_0}$ before entering R:

$$= \sum_{0 = i_0 < i_1 < \cdots < i_n \leq |w|} \Pr^w_A\big(\{\widetilde{q}_0 \widetilde{q}_1 \cdots \mid \{ i \mid \widetilde{q}_i \in \overline{Q_0} \} = \{i_0, \ldots, i_n\}, \; q_{i_n} \in P, \, \widetilde{q}_{i_n+1} \cdots \in R\}\big)$$

We apply Proposition 2.20 to move the first i_n positions out of $\Pr_{A'}$:

$$= \sum_{0 = i_0 < i_1 < \cdots < i_n \leq |w|} \sum_{\substack{\widetilde{q}_0, \ldots, \widetilde{q}_{i_n - 1} \in Q', \, \widetilde{q}_{i_n} \in \overline{P} \\ \{ i \mid \widetilde{q}_i = \overline{q}_i \} = \{i_0, \ldots, i_n\}}} \mu'(\widetilde{q}_0) \left(\prod_{i=1}^{i_n} \delta'(\widetilde{q}_{i-1}, w_i, \widetilde{q}_i) \right) \Pr^{w_{i_n+1} \cdots}_{A'_{\widetilde{q}_{i_n}}}(\widetilde{q}_{i_n} R)$$

Next, we insert the definition of δ':

$$= \sum_{0 = i_0 < i_1 < \cdots < i_n \leq |w|} \sum_{\substack{\overline{q}_{i_j} \in \overline{Q_0}, \, q_{i_j+1}, \ldots, q_{i_{j+1}-1} \in Q_0 \\ (j = 0, \ldots, n-1)}} \sum_{\overline{q}_{i_n} \in \overline{P}} \frac{\mu(q_0)}{1 - \mu(X_E)}$$
$$\prod_{j=1}^{n} \left[\left(\prod_{i=i_{j-1}+1}^{i_j - 1} \delta(q_{i-1}, w_i, q_i) \right) \left(\sum_{r \in X_E} \delta(q_{i_j - 1}, w_{i_j}, r) \frac{\mu(q_{i_j})}{1 - \mu(X_E)} \right) \right] \cdot \Pr^{w_{i_n+1} \cdots}_{A'_{\overline{q}_{i_n}}}(\overline{q}_{i_n} R)$$

The sums over $q_{i_j}, \dots, q_{i_{j+1}-1}$ for $j = 0, \dots, n-1$ are independent from each other. Thus, we can apply distributivity and move the sums into the product over the j's:

$$= \sum_{0=i_0<i_1<\cdots<i_n\le|w|} \prod_{j=1}^{n} \frac{1}{1-\mu(X_E)} \left(\sum_{\substack{q_{i_{j-1}},\dots,q_{i_j-1}\in Q_0 \\ q_{i_j}\in X_E}} \mu(q_{i_{j-1}}) \prod_{i=i_{j-1}+1}^{i_j} \delta(q_{i-1}, w_i, q_i) \right) \cdot \frac{1}{1-\mu(X_E)} \sum_{q\in P} \mu(q) \Pr_{A'_{\overline{q}}}^{w_{i_n+1}\cdots}(qR)$$

$$= \sum_{\substack{u_1\cdots u_n v=w \\ u_1,\dots,u_n\in\Sigma^+,\, v\in\Sigma^\infty}} \left(\prod_{j=1}^{n} \frac{\|A[X_E]_{\mathrm{F}}\|(u_i)}{1-\mu(X_E)} \right) \cdot \frac{1}{1-\mu(X_E)} \sum_{q\in P} \mu(q) \Pr_{A'_q}^{v}(qR).$$

This shows (6.3). Next, we apply (6.3) to the case where R is the set of all final states. Let $Y \subseteq Q_0$ be a set of states, let $\widetilde{Y} = Y \cup \overline{Y}$. We show

$$\|A'[\widetilde{Y}]_{\mathrm{F}}\| \equiv \|E\|^* \|A[Y]_{\mathrm{F}}\|. \tag{6.4}$$

Let $w \in \Sigma^*$. We obtain

$$\|A'[\widetilde{Y}]_{\mathrm{F}}\|(w) = \Pr_{A'}^{w}(Q'^* Y) + \Pr_{A'}^{w}(Q'^* \overline{Y})$$

We apply (6.3) twice: to the first summand with $R = Q_0^* Y$ and to the second summand with $R = \{\varepsilon\}$:

$$= \sum_{n\ge0} \sum_{\substack{u_1\cdots u_n v=w \\ u_1,\dots,u_n\in\Sigma^+,\, v\in\Sigma^*}} \left(\prod_{j=1}^{n} \frac{\|A[X_E]_{\mathrm{F}}\|(u_i)}{1-\mu(X_E)} \right) \sum_{q\in Q_0} \frac{\mu(q)}{1-\mu(X_E)} \Pr_{A'_q}^{v}(qQ_0^* Y) + \sum_{n\ge0} \sum_{\substack{u_1\cdots u_n v=w \\ u_1,\dots,u_n\in\Sigma^+,\, v\in\Sigma^*}} \left(\prod_{j=1}^{n} \frac{\|A[X_E]_{\mathrm{F}}\|(u_i)}{1-\mu(X_E)} \right) \sum_{q\in Y} \frac{\mu(q)}{1-\mu(X_E)} \Pr_{A'_q}^{v}(q)$$

Using the series expansion of $1/1-\mu(X_E)$ and that δ' and δ agree on runs only containing states from Q_0 we obtain:

$$= \sum_{n\ge0} \sum_{\substack{u_1\cdots u_n v=w \\ u_1,\dots,u_n,v\in\Sigma^+}} \sum_{\ell_0,\dots,\ell_n\in\mathbb{N}_0} \left(\prod_{j=1}^{n} \mu(X_E)^{\ell_{j-1}} \|A[X_E]_{\mathrm{F}}\|(u_i) \right) \mu(X_E)^{\ell_n} \Pr_{A}^{v}(Q_0^+ Y) + \sum_{n\ge0} \sum_{\substack{u_1\cdots u_n=w \\ u_1,\dots,u_n\in\Sigma^+}} \sum_{\ell_0,\dots,\ell_n\in\mathbb{N}_0} \left(\prod_{j=1}^{n} \mu(X_E)^{\ell_{j-1}} \|A[X_E]_{\mathrm{F}}\|(u_i) \right) \mu(X_E)^{\ell_n} \mu(Y)$$

Any sequence $v_1, \ldots, v_n$ of words can be bijectively mapped to a sequence $u_1, \ldots, u_k$ of non-empty words and a sequence $\ell_0, \ldots, \ell_n$ of non-negative integers counting the empty words between the non-empty ones. Moreover, we can rewrite the second sum by replacing $\mu(Y)$ with $\Pr_A^v(Y) = \mu(Y)\, \mathbb{1}_{\{\varepsilon\}}(v)$:

$$
\begin{aligned}
&= \sum_{n \geq 0} \sum_{\substack{u_1 \cdots u_n v = w \\ u_1, \ldots, u_n, v \in \Sigma^*}} \left(\prod_{j=1}^{n} \|A[X_E]_{\mathrm{F}}\|(u_i) \right) (\Pr_A^v(Q_0^+ Y) + \Pr_A^v(Y)) \\
&= (\|E\|^* \|A[Y]_{\mathrm{F}}\|)(w)\,.
\end{aligned}
$$

This shows (6.4). Let $E^*F' \in \mathrm{HT}(E^*F + E^\omega)$. By the above computation we obtain $\|A'[\widetilde{X_{F'}}]_{\mathrm{F}}\| \equiv \|E^*F'\|$. Thus, A' satisfies Definition 6.20 (2). By definition of δ', if $q \in Q_0$ is a sink state in A so is q and $\overline{q}$ in A'. Hence, the partition $X' = \widetilde{X_0} \cup \bigcup_{F' \in \mathrm{TT}(F)} \widetilde{X_{F'}}$ satisfies Definition 6.20 (3).

Next, we show Definition 6.20 (1). First, consider an infinite word $w \in \Sigma^\omega$. Note that the set of runs ρ with $\inf(\rho) \in \mathcal{R}$ and the set of runs ρ' with $\inf(\rho') \in R_1$ are disjoint. Thus $\|A\|(w) = \|A[\mathcal{R}]_{\mathcal{R}}\|(w) + \|A[\mathcal{R}_1]_{\mathcal{R}}\|(w)$. We compute $\|A[\mathcal{R}_1]_{\mathcal{R}}\|(w)$:

$$
\|A[\mathcal{R}_1]_{\mathcal{R}}\|(w) = \Pr_A^w(\{\widetilde{q}_0 \widetilde{q}_1 \cdots \mid \widetilde{q}_i = \overline{q}_i \text{ for infinitely many } i\})
$$

As $Q_0^* \overline{Q_0}$ is prefix-free, we have $\bigcap_{n \geq 1} (Q_0^* \overline{Q_0})^n Q'^\omega = \{\rho \in Q'^\omega \mid \inf(\rho) \cap \overline{Q_0} \neq \emptyset\}$. By the continuity of measures we conclude:

$$
= \lim_{n \to \infty} \Pr_A^w\Big(\big(\overline{Q}_0 Q_0^* \big)^n \overline{Q}_0 Q'^\omega \Big)
$$

We apply (6.3) with $P = \overline{Q_0}$ and $R = Q'^\omega$:

$$
= \lim_{n \to \infty} \sum_{\substack{u_1 \cdots u_n v = w \\ u_1, \ldots, u_n \in \Sigma^+,\, v \in \Sigma^\omega}} \prod_{i=1}^{n} \frac{\|A[X_E]_{\mathrm{F}}\|(u_i)}{1 - \mu(X_E)}. \tag{6.5}
$$

Next, we derive that $\|E\|^\omega$ is equal to the expression (6.5).

$$
\begin{aligned}
\|E\|^\omega(w) &= \lim_{n \to \infty} \sum_{\substack{u_1 \cdots u_n v = w \\ u_1, \ldots, u_n \in \Sigma^*,\, v \in \Sigma^\omega}} \prod_{i=1}^{n} \|E\|(u_i) \\
&= \lim_{n \to \infty} \sum_{k=0}^{n} \sum_{\substack{u_1 \cdots u_n v = w \\ u_1, \ldots, u_n \in \Sigma^*,\, v \in \Sigma^\omega \\ |\{i \mid u_i \neq \varepsilon\}| = k}} \prod_{i=1}^{n} \|E\|(u_i)
\end{aligned}
$$

There are n over k combinations to choose k non-empty words from n possible words:

$$= \lim_{n\to\infty} \sum_{k=0}^{n} \sum_{\substack{u_1\cdots u_k v = w \\ u_1,\ldots,u_k \in \Sigma^+,\, v\in\Sigma^\omega}} \binom{n}{k} (\|E\|(\varepsilon))^{n-k} \prod_{i=1}^{k} \|E\|(u_i)$$

$$= \lim_{n\to\infty} \sum_{k=0}^{n} \underbrace{\left(\sum_{\substack{u_1\cdots u_k v = w \\ u_1,\ldots,u_k \in \Sigma^+,\, v\in\Sigma^\omega}} \prod_{i=1}^{k} \frac{\|E\|(u_i)}{1-\|E\|(\varepsilon)} \right)}_{= a_k} \binom{n}{k} (\|E\|(\varepsilon))^{n-k} (1-\|E\|(\varepsilon))^{k} \tag{6.6}$$

Now, we analyse the parenthesised expression a_k and show that a_k converges for $k \to \infty$. By assumption we have $\|E\| = \|A[X_E]_{\mathrm{F}}\|$. As every state in X_E is a sink state, $\|E\|$ is prefix summable. Thus, so is the series S given by $S(w) = \frac{\|E\|(w)}{1-\|E\|(\varepsilon)}$ for $w \neq \epsilon$ and $S(\varepsilon) = 0$. We can write $a_k = (S^k \cdot \mathbb{1})(w)$. By Lemma 6.7 $S^k \cdot \mathbb{1}$ is bounded by 1. Since series concatenation is monotonic, we obtain $a_k = (S^k \cdot \mathbb{1})(w) \geq (S^k \cdot (S \cdot \mathbb{1}))(w) = (S^{k+1} \cdot \mathbb{1})(w) = a_{k+1} \geq 0$. Hence, the sequence $(a_k)_{k\geq 0}$ is monotonic and bounded, and thus convergent. Therefore, we can apply Corollary 6.26, where every X_n is distributed as a binomial distribution with parameters n and $1 - \|E\|(\varepsilon)$, i.e., $\mathbb{E}[X_n] = n(1-\|E\|(\varepsilon))$ and $\sigma(X_n) = \sqrt{n(1-\|E\|(\varepsilon))\|E\|(\varepsilon)}$. As $\|E\|(\varepsilon) < 1$ by assumption, this choice of the X_n satisfies the requirements of Corollary 6.26. We obtain

$$(6.6) = \lim_{n\to\infty} \sum_{\substack{u_1\cdots u_n v = w \\ u_1,\ldots,u_n \in \Sigma^+,\, v\in\Sigma^\omega}} \prod_{i=1}^{n} \frac{\|E\|(u_i)}{1-\|E\|(\varepsilon)}. \tag{6.7}$$

Since $\|E\| = \|A[X_E]_{\mathrm{F}}\|$ by assumption, we have that (6.5) and (6.7) equal to each other, and so $\|E^\omega\|(w) = \|A[\mathcal{R}_1]_{\mathrm{F}}\|(w)$.

We show $\|A[\mathcal{R}]_{\mathcal{R}}\|(w) = \sum_{F'\in \mathrm{TT}(F)} \|E^* F\|(w)$. Note that any run $\rho \in Q'^\omega$ with $\inf(\rho) \in \mathcal{R}$ can only contain finitely many states from $\overline{Q_0}$. Therefore

$$\begin{aligned} \|A[\mathcal{R}]_{\mathcal{R}}\|(w) &= \Pr_A^w(\{\rho \in Q'^\omega \mid \inf(\rho) \in \mathcal{R}\}) \\ &= \Pr_A^w\Big((\overline{Q_0}Q_0^*)^*\overline{Q_0}R\Big) \text{ where } R = \{\rho \in Q_0^\omega \mid \inf(\rho) \in \mathcal{R}\} \end{aligned}$$

Since $(\overline{Q_0}Q_0^*)^*\overline{Q_0}R = \bigcup_{n\geq 0}(\overline{Q_0}Q_0^*)^n\overline{Q_0}R$ and the sets $(\overline{Q_0}Q_0^*)^n\overline{Q_0}R$ are pairwise disjoint for different values of n, we conclude

$$= \sum_{n\geq 0} \Pr_A^w\Big((\overline{Q_0}Q_0^*)^n\overline{Q_0}R\Big)$$

By (6.3) we obtain:

$$= \sum_{n\geq 0} \sum_{\substack{u_1\cdots u_n v = w \\ u_1,\ldots,u_n \in \Sigma^+,\, v\in\Sigma^\omega}} \left(\prod_{i=1}^{n} \frac{\|A[X_E]_{\mathrm{F}}\|(u_i)}{1-\mu(X_E)}\right) \sum_{q\in Q_0} \frac{\mu(q)}{1-\mu(X_E)} \Pr^{v}_{A'_q}(qR)$$

As A and A' agree on runs from Q_0^ω, X_E only contain sink states, and $R = Q_0 R$:

$$= \sum_{n\geq 0} \sum_{\substack{u_1\cdots u_n v = w \\ u_1,\ldots,u_n \in \Sigma^+,\, v\in\Sigma^\omega}} \left(\prod_{i=1}^{n} \frac{\|E\|(u_i)}{1-\|E\|(\varepsilon)}\right) \frac{1}{1-\|E\|(\varepsilon)} \Pr^{v}_{A}(R)$$

As before we move from sum over non-empty words, to the sum over all words by expanding the geometric series:

$$\begin{aligned} &= \sum_{n\geq 0} \sum_{\substack{u_1\cdots u_n v = w \\ u_1,\ldots,u_n \in \Sigma^*,\, v\in\Sigma^\omega}} \left(\prod_{i=1}^{n} \|E\|(u_i)\right) \sum_{F'\in \mathrm{TT}(F)} \|F'\|(v) \\ &= \sum_{F'\in \mathrm{TT}(F)} \|E^*F\|(w)\,. \end{aligned}$$

Now, assume $w \in \Sigma^*$ is a finite word. By (6.4) we have

$$\begin{aligned} \|A[\widetilde{X_0}]_{\mathrm{F}}\|(w) = (\|E\|^* \cdot \|A[X_0]_{\mathrm{F}}\|)(w) &= \sum_{F'\in \mathrm{TT}(F)} (\|E\|^* \|F'\|)(w) \\ &= \sum_{H\in \mathrm{TT}(E^*F)} \|H\|(w). \end{aligned}$$

This completes the proof. We have shown that A' is an automaton for $E^*F + E^\omega$. The cases E^*F and E^ω are completely analogous: in the first case omit all repeated states in A', and in the second state omit all final states and include only $\mathcal{R}_1$ as repeated states. ∎

Corollary 6.28. Let E be a PRE. Then, there is an automaton for E.

Proof. By Lemmas 6.21 to 6.24 and 6.27 the set of all expressions E, which admit an automaton for E, satisfies the closure conditions of Definition 6.12. Therefore, this set already contains all probabilistic regular expressions. ∎

6.5 From Automata to Expressions

In this section, we show that there is an equivalent probabilistic regular expression for every probabilistic Muller-automaton. In contrast to [BGMZ12], where this step of the proof resembles Kleene's classical proof, we use a different construction in

order to capture the Muller-acceptance condition. The main tool in our proof are prefix-free sets of runs. These allow us to uniquely decompose runs which arise from concatenation or iteration of prefix-free sets of runs.

6.5.1 Prefix-free sets of runs

The following definition is folklore, but repeated here for completeness.

Definition 6.29. Let Q be an alphabet and $L \subseteq Q^*$.

1. The set of all *prefixes of* L is defined by
$$\mathrm{Pre}(L) = \{u \in Q^* \mid \exists v \in Q^* : uv \in L\}.$$
2. The set L is *prefix-free* if for every $w \in L$ we have
$$\mathrm{Pre}(\{w\}) \cap L = \{w\}.$$

Concatenation and iteration of prefix-free languages removes the ambiguity from these operations and yields unique decompositions.

Proposition 6.30. Let $L \subseteq Q^*$ be prefix-free and $K \subseteq Q^\infty$ with $L^+K \cap K = \emptyset$. Then

1. If $w \in LQ^\infty$, then $w = uv$ for unique $u \in L$ and $v \in Q^\infty$.
2. If $w \in L^*K$, then $w = u_1 \cdots u_n v$ for unique a $n \geq 0$ and $u_1, \ldots, u_n \in L$, $v \in K$.

Proof. **1.** Assume $uv = u'v'$ with $u, u' \in L$ and $v, v' \in Q^\infty$. if $|u| < |u'|$, then u is a strict prefix of u'. This contradicts the prefix-freeness of L. Analogously, $|u| > |u'|$ is not possible. Thus, $u = u'$ and $v = v'$.

2. Consider words $u_1 \cdots u_n v = u'_1 \cdots u'_m v'$ with $u_i, u'_i \in L$ and $v, v' \in K$. Assume an $i \leq \min(n, m)$ with $u_i \neq u'_i$. Let i be minimal with this property. Thus, $u_i \cdots u_n v = u'_i \cdots u'_m v \in LQ^*$. By 1. we have $u_i = u'_i$. A contradiction. Hence, $u_i = u'_i$ for all $i \leq \min(n, m)$. Next, assume $n < m$, i.e., $v = u'_{n+1} \cdots u'_m v' \in K \cap L^+K$. This contradicts the assumption $L^+K \cap K = \emptyset$. Therefore, $m = n$ and so $v = v'$. ∎

Lemma 6.31. Let $L \subseteq Q^+$ be prefix-free. Then $L^\omega = \bigcap_{n \geq 0} L^n Q^\omega$.

Proof. The direction "$\subseteq$" is clear. Let $w \in L^n Q^\omega$ for all $n \geq 0$. For any $n \geq 0$ there are words $w_1^{(n)}, \ldots, w_n^{(n)} \in L$ and $v^{(n)} \in Q^\omega$ with $w = w_1^{(n)} \cdots w_n^{(n)} v^{(n)}$. Let $0 \leq n \leq m$. We have $w = w_1^{(n)} \cdots w_n^{(n)} v^{(n)} = w_1^{(m)} \cdots w_n^{(m)} (w_{n+1}^{(m)} \cdots w_m^{(m)} v^{(m)})$. Hence, by Proposition 6.30, we obtain $w_i^{(n)} = w_i^{(m)}$ for all $1 \leq i \leq n$. Thus, the word $w_1^{(1)} w_2^{(2)} \cdots$ equals w and we have $w \in L^\omega$. ∎

Note that $L^\omega = \bigcap_{n\geq 0} L^n Q^\omega$ does not hold for general languages L. Consider $Q = \{\mathrm{a}, \mathrm{b}\}$ and the sequence $(w_i)_{i\geq 0}$ of words defined by $w_0 = \mathrm{a}$ and $w_{n+1} = w_n^{n+1}\mathrm{b}$ for $n \geq 0$, i.e., $w_1 = \mathrm{ab}$, $w_2 = \mathrm{ababb}$ and so on. As w_n is a strict prefix of w_{n+1}, there is a word $w \in Q^\omega$ such that every w_n is a prefix of w. Let $L = \{w_n \mid n \geq 0\}$. Clearly, $w \in L^n Q^\omega$ for every $n \geq 0$, but $w \notin L^\omega$.

Using the results of Proposition 6.30, we can decompose the probability of concatenation and iteration of prefix-free sets of runs.

Lemma 6.32. Let $A = (Q, \delta, \mu, F, \mathcal{R})$ be a probabilistic Muller-automaton, $q \in Q$, $L \subseteq Q^+$ such that Lq is prefix-free, and $K \subseteq Q^\infty$. Then

1. $\Pr_A(LqK) = \Pr_A(Lq) \cdot \Pr_{A_q}(qK)$,
2. If $(Lq)^+K \cap K = \emptyset$ then $\Pr_{A_q}(q(Lq)^*K) = \big(\Pr_{A_q}(qLq)\big)^* \cdot \Pr_{A_q}(qK)$,
3. $\Pr_{A_q}(q(Lq)^\omega) = \big(\Pr_{A_q}(qLq)\big)^\omega$,

where $A_q = (Q, \delta, \mathbb{1}_{\{q\}}, F, \mathcal{R})$ for all $q \in Q$.

Proof. **1.** Let $w = w_1 \cdots w_n \in LqK$. By Proposition 6.30 there is a unique $k \geq 1$ such that $w_1 \cdots w_k \in Lq$ and $w_{k+1} \cdots w_n \in K$. Thus, the sets $(Lq \cap Q^n)K$ are pairwise disjoint. We conclude

$$\begin{aligned}\Pr_A^w(LqK) &= \sum_{n\geq 1} \Pr_A^w((Lq \cap Q^n)K) \\ &= \sum_{n\geq 1} \Pr_A^{w_1\cdots w_{n-1}}(Lq \cap Q^n) \Pr_{A_q}^{w_n\cdots}(qK) \\ &= \big(\Pr_A(Lq) \cdot \Pr_{A_q}(qK)\big)(w).\end{aligned}$$

2. Again, by Proposition 6.30, the sets $(Lq)^nK$ are pairwise disjoint. Hence

$$\begin{aligned}\Pr_{A_q}^w(q(Lq)^*K) = \sum_{n\geq 0} \Pr_{A_q}^w(q(Lq)^nK) &= \sum_{n\geq 0} \Big(\Big(\Pr_{A_q}(qLq)\Big)^n \cdot \Pr_{A_q}.(qK)\Big)(w) \\ &= \big(\Pr_{A_q}(qLq)\big)^*(w)\end{aligned}$$

3. We apply the result from 1. and the definition of infinity iteration. Thus

$$\begin{aligned}\Pr_{A_q}^w(q(Lq)^\omega) = \lim_{n\to\infty} \Pr_{A_q}^w(q(Lq)^nQ^\omega) &= \lim_{n\to\infty} \big(\big(\Pr_{A_q}(qLq)\big)^n \cdot \Pr_{A_q}(qQ^\omega)\big)(w) \\ &= \lim_{n\to\infty} \big(\big(\Pr_A(qLq)\big)^n \cdot \mathbb{1}\big)(w) = \big(\Pr_A(qLq)\big)^\omega(w).\end{aligned}$$

∎

In order to obtain an inductive proof, based on the number of visited states, we next show how the Kleene-iteration and ω-iteration of a set can be decomposed into iterations of smaller, prefix-free sets. This lemma will be the major ingredient for the construction of an equivalent probabilistic regular expression for a probabilistic Muller-automaton.

For any two sets of states $F, X \subseteq Q$ let

$$R_F^X = \{\rho \in X^\omega \mid \inf(\rho) = F\}.$$

Lemma 6.33. Let Q be a finite, non-empty set, $\emptyset \neq X \subseteq Q$ a subset of Q, and $X = \{x_1, \ldots, x_m\}$ an enumeration of X. Furthermore, let $\emptyset \neq F \subsetneq X$. We define the following sets C_k for $0 \leq k \leq m$:

$$C_k = \begin{cases} X_1^* x_1 X_2^* x_2 \cdots x_{k-1} X_k^* x_k & \text{if } k > 0 \\ \{\varepsilon\} & \text{if } k = 0, \end{cases}$$

where $X_i = X \setminus \{x_i\}$. The following equalities hold:

$$X^* = \bigcup_{k=1}^{m} (C_m)^* \cdot C_{k-1} \cdot X_k^*, \tag{6.8}$$

$$R_F^X = \bigcup_{k=1}^{m} (C_m)^* \cdot C_{k-1} \cdot R_F^{X_k}, \tag{6.9}$$

$$R_X^X = (C_m)^\omega, \tag{6.10}$$

Moreover, the unions in the first and second equation are over pairwise disjoint sets.

Proof. We show the (6.8). As $x_i \in X$ and $X_i \subseteq X$, the direction "$\supseteq$" is clear. Let $\phi\colon \mathbb{N} \to \{1, \ldots, m\}$ map a positive integer n to the positive remainder when divided by m, i.e., we have $n = a \cdot m + \phi(n)$ for some $a \geq 0$ and all $n > 0$.

Let $w \in X^*$. We inductively define a sequence $(n_i)_{i \geq 0}$ of non-negative integers by $n_0 = 0$ and

$$n_i = \min\big(\{k \in \mathrm{pos}(w) \mid k > n_{i-1},\ w_k = x_{\phi(i)}\} \cup \{\infty\}\big)$$

for all $i > 0$. Note that $n_{i+1} > n_i$ if $n_i < \infty$. Let $N = \min\{i \mid n_i = \infty\}$ and define

$$\begin{aligned} u_i &= w_{n_{i-1}+1} \cdots w_{n_i} && \text{for all } 1 \leq i < N, \\ u_N &= w_{n_{N-1}+1} \cdots w_{|w|}. \end{aligned}$$

We have $w = u_1 \cdots u_N$, and for $i < N$ it holds that

$$u_i \in X^*_{\phi(i)} x_{\phi(i)} \qquad \text{and} \qquad u_N \in X^*_{\phi(N)}.$$

Thus, for every $j \geq 0$ of m we have $u_{mj+1} \cdots u_{mj+m} \in C_m^X$. Let $N = a \cdot m + b$ with $1 \leq b \leq m$ and $a \geq 0$. We obtain

$$w = u_1 \ldots u_N = \left(\prod_{j=0}^{a-1} u_{mj+1} \cdots u_{mj+m} \right) \left(u_{ma+1} \cdots u_{ma+(b-1)} \right) u_{ma+b}.$$

Thus, $w \in (C_m)^* C_{\phi(N)-1} X^*_{\phi(N)}$.

We show that sets of the form $(C_m)^* C_{k-1} L_k$, where $L_k \subseteq X_k^\infty$, are pairwise disjoint for different values of k. Let $k, k' \in \{1, \ldots, m\}$ and assume $w \in (C_m)^* C_{k-1} L_k \cap (C_m)^* C_{k'-1} L_{k'}$. Let $w = u_1 \cdots u_n v z = u'_1 \cdots u'_{n'} v' z'$ with $u_1, \ldots, u_n, u'_1, \ldots, u'_{n'} \in C_m$, $v \in C_{k-1}$, $v' \in C_{k'-1}$, $z \in L_k$, and $z' \in L_{k'}$.

Assume $n < n'$. Hence, $vz = u'_{n+1} \cdots u'_{n'} v' z'$. As $u'_{n+1} \in C_m$, it contains a prefix p in $C_k = C_{k-1} X_k^* x_k$, i.e., $p = p_1 p_2 x_k$ with $p_1 \in C_{k-1}$ and $p_2 \in X_k^*$. By Proposition 6.30 we have $p_1 = v$. Thus, $p_2 x_k$ is a prefix of z. But z does not contain the symbol x_k, so $p_2 x_k$ cannot be a prefix of z. A contradiction. Analogously $n > n'$ is not possible. Thus $n = n'$ and $u_i = u'_i$ for all $1 \leq i \leq n$ as C_m is prefix-free by Proposition 6.30. Hence, $vz = v'z'$. If $k < k'$ then vz would have a prefix in $C_{k'}$, which results in a contradiction as before. Similarly, $k > k'$ is not possible. Therefore, $k = k'$. This shows that the unions in (6.9) and (6.10) are unions of pairwise disjoint sets.

Next, we consider an infinite word $w \in \Sigma^\omega$ and show (6.9) and (6.10). In both cases the direction "$\supseteq$" is clear. We will use the sequence $(n_i)_{i \geq 0}$ as defined above. We show (6.9). Let $w \in R_F^X$ with $F \subsetneq X$. Thus, there is a $x \in X$ which occurs only finitely often in w. Hence, we have $n_i = \infty$ for some $i \geq 0$. Let N be the minimal index i with $n_i = \infty$. We define words u_i as in the previous case. As w is infinite, we have $u_N \in X^\omega_{\phi(N)}$. Furthermore, $\inf(u_N) = \inf(w) = F$. Thus $u_N \in R_F^{X_{\phi(N)}}$. As before, we obtain $w \in (C_m)^* C_{\phi(N)-1} R_F^{X_{\phi(N)}}$.

We show (6.10). Let $w \in R_X^X$. As every $x_k \in X$ occurs infinitely often in w, we have $n_i < \infty$ for all $i \geq 0$. Using the words u_i from the previous two cases, we have $u_i \in X_{\phi(i)} x_{\phi(i)}$ for all $i \geq 1$ and $w = u_1 u_2 \cdots = \prod_{j \geq 0} (u_{jm+1} \cdots u_{jm+m}) \in (C_m^X)^\omega$. ∎

6.5.2 Constructing an Expression for an Automaton

The next lemma is an extension of the syntax rule Definition 6.12 (5), which allows an additional test for the empty word.

Lemma 6.34. If $\varepsilon + E\Box + F$ is a PRE, then there are expressions $\widetilde{E^*}$, $\widetilde{E^*}$, and $\widetilde{E^\omega}$ such that $\widetilde{E^*} \equiv E^*$, $\widetilde{E^+} \equiv E^+$, $\widetilde{E^\omega} \equiv E^\omega$, and $\varepsilon + \widetilde{E^+} + \widetilde{E^\omega} + \widetilde{E^*} F$ is also an expression.

Proof. By Definition 6.12 (5), we have that $E^* + E^*F + E^\omega$ is an expression. We substitute this expression in $\varepsilon + E\Box + F$ using Lemma 6.18 and obtain that

$$\varepsilon + E\big(E^* + E^*F + E^\omega\big) + F \equiv \varepsilon + EE^* + (EE^* + \varepsilon)F + EE^\omega$$

is a probabilistic regular expression. Setting $\widetilde{E^+} = EE^*$, $\widetilde{E^*} = \varepsilon + EE^*$, and $\widetilde{E^\omega} = EE^\omega$ completes the proof. ∎

Lemma 6.35. Let A be a probabilistic Muller-automaton. There is a complete PRE E with $\|A\| = \|E\|$.

Proof. Let $A = (Q, \delta, \mu, F, \mathcal{R})$. Given a set $X \subseteq Q$ and a state $p \in Q$ such that either $p \in X$ or $X = \emptyset$, we construct expressions E_p^X of the following form:

$$E_p^X = \varepsilon + \sum_{q \in Q \setminus X} E_{p,q}^X \Box + \sum_{q \in X} E_{p,q}^X + \sum_{\emptyset \neq R \subseteq X} E_{p,\inf=R}^X, \tag{6.11}$$

where the subexpressions $E_{p,q}^X$ and $E_{p,\inf=R}^X$ have the following semantics

$$\left\|E_{p,q}^X\right\| = \Pr_{A_p}(pX^*q), \tag{6.12}$$

$$\left\|E_{p,\inf=R}^X\right\| = \Pr_{A_p}\big(\{\rho \in X^\omega \mid \inf(\rho) = R\}\big). \tag{6.13}$$

We use induction on $|X|$. For $X = \emptyset$, we have $\|E_{p,q}^\emptyset\| = \|\sum_{a \in \Sigma} \delta(p, a, q)\, a\|$. We construct an expression $E_p^\emptyset$ using Definition 6.12 (1) to Definition 6.12 (3), distributivity, and associativity:

$$E_p^\emptyset = \varepsilon + \sum_{q \in Q}\left(\sum_{a \in \Sigma} a\, \delta(p, a, q)\right)\Box = \varepsilon + \sum_{a \in \Sigma} a\left(\sum_{q \in Q} \delta(p, a, q)\Box\right).$$

Assume $X \neq \emptyset$ and let $p \in X$. Fix an enumeration $\{x_1, \ldots, x_m\}$ with $x_m = p$ of X and let $X_i = X \setminus \{x_i\}$. Furthermore, let $x_0 = x_m$. By induction hypothesis, there are expressions

$$E_{x_i}^{X_{i+1}} = \varepsilon + E_{x_i,x_{i+1}}^{X_{i+1}}\Box + \sum_{q \in Q \setminus X} E_{x_i,q}^{X_{i+1}}\Box + \sum_{q \in X_{i+1}} E_{x_i,q}^{X_{i+1}} + \sum_{\emptyset \neq R \subseteq X_{i+1}} E_{x_i,\inf=R}^{X_{i+1}},$$

for every $i = 0, \ldots, m-1$. We show that for every $k = 0, \ldots, m-1$, the following expression E'_k is a probabilistic regular expression:

$$E'_k = \varepsilon + \mathcal{C}_{k+1}\Box + \sum_{q \in Q \setminus X} \sum_{i=0}^{k} \mathcal{C}_i E_{x_i,q}^{X_{i+1}}\Box + \sum_{q \in X} \sum_{i=0}^{k-1} \mathcal{C}_i E_{x_i,q}^{X_{i+1}} + \sum_{q \in X_{k+1}} \mathcal{C}_k E_{x_k,q}^{X_{k+1}}$$

$$+ \sum_{\emptyset \neq R \subseteq X} \sum_{\substack{i \in \{0,\dots,k-1\} \\ x_{i+1} \notin R}} \mathcal{C}_i E^{X_{k+1}}_{x_i,\inf=R},$$

where we set $\mathcal{C}_i = E^{X_1}_{x_0,x_1} E^{X_2}_{x_1,x_2} \cdots E^{X_{i+1}}_{x_{i-1},x_i}$ for $i > 0$, and $\mathcal{C}_0$ as the empty expression. For $k = 0$, E'_k equals to $E^{X_1}_{x_0,x_1}$. Assume that E'_k is a probabilistic regular expression for $k < m-1$, we show that the same holds for E'_{k+1}. Using Lemma 6.18, we substitute the $\square$ after $\mathcal{C}_{k+1}$ in E'_k by $E^{X_{k+2}}_{x_{k+1}}$ resulting in the following expression:

$$\begin{aligned} E'' = \varepsilon + \mathcal{C}_{k+1} \Bigg(& \varepsilon + E^{X_{k+2}}_{x_{k+1},x_{k+2}} \square + \sum_{q \in Q \setminus X} E^{X_{k+2}}_{x_{k+1},q} \square + \sum_{q \in X_{k+2}} E^{X_{k+2}}_{x_{k+1},q} + \sum_{\emptyset \neq R \subseteq X_{k+2}} E^{X_{k+2}}_{x_{k+1},\inf=R} \Bigg) \\ & + \sum_{q \in Q \setminus X} \sum_{i=0}^{k} \mathcal{C}_i E^{X_{i+1}}_{x_i,q} \square + \sum_{q \in X} \sum_{i=0}^{k-1} \mathcal{C}_i E^{X_{i+1}}_{x_i,q} + \sum_{q \in X_{k+1}} \mathcal{C}_k E^{X_{k+1}}_{x_k,q} \\ & + \sum_{\emptyset \neq R \subseteq X} \sum_{\substack{i \in \{0,\dots,k-1\} \\ x_{i+1} \notin R}} \mathcal{C}_i E^{X_{k+1}}_{x_i,\inf=R}, \end{aligned}$$

Using associativity, commutativity, and distributivity, we obtain

$$\begin{aligned} = \varepsilon + \mathcal{C}_{k+1} E^{X_{k+2}}_{x_{k+1},x_{k+2}} \square + \sum_{q \in Q \setminus X} \Bigg(& \mathcal{C}_{k+1} E^{X_{k+2}}_{x_{k+1},q} \square + \sum_{i=0}^{k} \mathcal{C}_i E^{X_{i+1}}_{x_i,q} \square \Bigg) \\ & + \underbrace{\sum_{q \in X} \sum_{i=0}^{k-1} \mathcal{C}_i E^{X_{i+1}}_{x_i,q} + \sum_{q \in X_{k+1}} \mathcal{C}_k E^{X_{k+1}}_{x_k,q} + \mathcal{C}_{k+1}}_{= \sum_{q \in Q} \sum_{i=0}^{k} \mathcal{C}_i E^{X_{i+1}}_{x_i,q}} + \sum_{q \in X_{k+2}} \mathcal{C}_{k+1} E^{X_{k+2}}_{x_{k+1},q} \\ & + \sum_{\emptyset \neq R \subseteq X} \sum_{\substack{i \in \{0,\dots,k-1\} \\ x_{i+1} \notin R}} \mathcal{C}_i E^{X_{k+1}}_{x_i,\inf=R} + \sum_{\emptyset \neq R \subseteq X} \sum_{\substack{i=k+1 \\ x_{i+1} \notin R}} \mathcal{C}_i E^{X_{i+1}}_{x_i,\inf=R}. \end{aligned}$$

This expression is equal to E'_{k+1}. Hence, we obtain that E'_k is a probabilistic regular expression for all $k = 0, \dots, m-1$. In particular, the expression E'_{m-1} is of the following form:

$$\begin{aligned} E'_{m-1} = \varepsilon + \mathcal{C}_m \square + \sum_{q \in Q \setminus X} \sum_{i=0}^{m-1} \mathcal{C}_i E^{X_{i+1}}_{x_i,q} \square + \sum_{q \in X} \sum_{i=0}^{m-2} \mathcal{C}_i E^{X_{i+1}}_{x_i,q} + \sum_{q \in X_m} \mathcal{C}_{m-1} E^{X_m}_{x_{m-1},q} \\ + \sum_{\emptyset \neq R \subseteq X} \sum_{\substack{i \in \{0,\dots,m-2\} \\ x_{i+1} \notin R}} \mathcal{C}_i E^{X_{m-1}}_{x_i,\inf=R}, \end{aligned}$$

Next, we apply Lemma 6.34 iterating $\mathcal{C}_m$ and obtain the following expression E''':

$$E''' = \varepsilon + \mathcal{C}_m^+ + \sum_{q\in Q\setminus X}\sum_{i=0}^{m-1} \mathcal{C}_m^*\mathcal{C}_i E_{x_i,q}^{X_{i+1}}\square + \sum_{q\in X}\sum_{i=0}^{m-2} \mathcal{C}_m^*\mathcal{C}_i E_{x_i,q}^{X_{i+1}} + \sum_{q\in X_m} \mathcal{C}_m^*\mathcal{C}_{m-1}E_{x_{m-1},q}^{X_m} + \sum_{\emptyset\neq R\subsetneq X}\sum_{\substack{i=0\\ x_{i+1}\notin R}}^{m-1} \mathcal{C}_m^*\mathcal{C}_i E_{x_i,\inf=R}^{X_{i+1}} + \mathcal{C}_m^\omega,$$

where we used the expressions $\widetilde{\mathcal{C}_m^+}$, $\widetilde{\mathcal{C}_m^*}$, $\widetilde{\mathcal{C}_m^\omega}$ from Lemma 6.34 without the tilde to increase the readability of the formula.

We define the expressions $E_{x_0,q}^X$, $E_{x_0,\inf=R}^X$, and $E_{x_0,\inf=X}^X$ for every state $q \in Q$ and subset $\emptyset \neq R \subsetneq X$ by

$$E_{x_0,q}^X = \begin{cases} \sum_{i=0}^{m-1} \mathcal{C}_m^*\mathcal{C}_i E_{x_i,q}^{X_{i+1}} & \text{if } q \neq x_m \\ \sum_{i=0}^{m-2} \mathcal{C}_m^*\mathcal{C}_i E_{x_i,q}^{X_{i+1}} + \mathcal{C}_m^+ & \text{if } q = x_m, \end{cases}$$

$$E_{x_0,\inf=R}^X = \sum_{\substack{i=0\\ x_{i+1}\notin R}}^{m-1} \mathcal{C}_m^*\mathcal{C}_i E_{x_i,\inf=R}^{X_{i+1}},$$

$$E_{x_0,\inf=X}^X = \mathcal{C}_m^\omega.$$

Thus, by using commutativity and associativity, we obtain that E''' is in the form of (6.11).

We still need to show that just defined expressions satisfy the semantics properties (6.12) and (6.13). We first show (6.12): since $\mathcal{C}_m^+ \equiv \mathcal{C}_m^*\mathcal{C}_{m-1}E_{x_{m-1},x_m}^{X_m}$, we can assume that $E_{x_0,q}^X = \sum_{i=0}^{m-1} \mathcal{C}_m^*\mathcal{C}_i E_{x_i,q}^{X_{i+1}}$ for all $q \in Q$. Let $q \in Q$.

$$\left\|E_{x_0,q}^X\right\| = \sum_{i=0}^{m-1}\left(\left\|E_{x_0,x_1}^{X_1}\right\|\cdots\left\|E_{x_{m-1},x_m}^{X_m}\right\|\right)^*\left\|E_{x_0,x_1}^{X_1}\right\|\cdots\left\|E_{x_i,q}^{X_{i+1}}\right\|$$

By induction hypothesis, we obtain

$$= \sum_{i=0}^{m-1}\Big(\Pr_{A_{x_0}}(x_0X_1^*x_1)\cdots\Pr_{A_{x_{m-1}}}(x_{m-1}X_m^*x_m)\Big)^* \cdot \Pr_{A_{x_0}}(x_0X_1^*x_1)\cdots\Pr_{A_{x_{i-1}}}(x_{i-1}X_i^*x_i)\Pr_{A_{x_i}}(x_iX_{i+1}^*q)$$

By Lemma 6.32 we move concatenation and iteration in the probability measure

$$\overset{*}{=} \sum_{i=0}^{m-1}\Big(\Pr_{A_{x_0}}\big(x_0X_1^*x_1\cdots x_{m-1}X_m^*x_m\big)\Big)^*\Pr_{A_{x_0}}\big(x_0X_1^*x_1\cdots x_iX_{i+1}^*q\big)$$

$$\overset{*}{=} \sum_{i=0}^{m-1} \Pr_{A_{x_0}}\Big(x_0\big(X_1^* x_1 X_2^* \cdots X_{m-1}^* x_m\big)^* X_1^* x_1 X_2^* \cdots X_i^* x_i X_{i+1}^* q\Big)$$

Finally, we apply (6.8) from Lemma 6.33 to obtain

$$= \Pr_{A_{x_0}}(x_0 X^* q).$$

The requirements of Lemma 6.32 in the two equalities $*$ are satisfied: Let $C_i = X_1^* x_1 \cdots x_{i-1} X_i^* x_i$. We show $C_m^+ C_i X_{i+1}^\infty \cap C_i X_{i+1}^\infty = \emptyset$ for every $0 \le i \le m-1$. Assume $w \in C_m^+ C_i X_{i+1}^* \cap C_i X_{i+1}^\infty$, i.e., $w = u_1 x_1 u_2 x_2 \cdots u_m x_m v$ and $w = u_1' x_1 \cdots u_i' x_i v'$ for $u_i, u_i' \in X_i^*$, $v \in Q^+$, and $v' \in X_{i+1}^\infty$. By Proposition 6.30 we have $u_i = u_i'$ for $i \le i$. Thus, $u_{i+1} x_{i+1} \cdots u_m x_1 v = v'$. This is a contradiction, as x_{i+1} occurs on the left side of the equation, but not on the right side. Therefore, $C_m^+ C_i X_{i+1}^\infty \cap C_i X_{i+1}^\infty = \emptyset$. Hence, $C_m^+ K \cap K = \emptyset$ and so $C_m^+ K q \cap K q = \emptyset$ for all $K \subseteq C_i X_{i+1}^\omega$ and $q \in Q$. Therefore, we can apply Lemma 6.32 and we obtain (6.12).

For (6.13) first consider the case $R \neq X$. The calculation is essentially the same as in the previous case.

$$\begin{aligned}\|E^X_{x_0,\inf=R}\| = \sum_{i=0}^{m=1} &\big(\Pr_{A_{x_0}}(x_0 X_1^* x_1) \cdots \Pr_{A_{x_{m-1}}}(x_{m-1} X_m^* x_m)\big) \\ &\cdot \Pr_{A_{x_0}}(x_0 X_1^* x_1) \cdots \Pr_{A_{x_{i-1}}}(x_{i-1} X_i^* x_i) \\ &\cdot \Pr_{A_{x_i}}(\{\rho \in X_{i+1}^\omega \mid \inf(\rho) = R\})\end{aligned}$$

As before, we apply Lemma 6.32:

$$= \sum_{i=0}^{m-1} \Pr_{A_{x_0}}\big(x_0 C_m^* C_i \{\rho \in X_{i+1}^\omega \mid \inf(\rho) = R\}\big)$$

By Lemma 6.33 we obtain

$$= \Pr_{A_{x_0}}\big(\{\rho \in X^\omega \mid \inf(\rho) = R\}\big).$$

Finally, consider the case $R = X$. We defined $E^X_{x_0,\inf=X} = (\mathcal{C}_m)^\omega$. By Lemma 6.32 we know $\|\mathcal{C}_m^\omega\| = \Big(\Pr_{A_{x_0}}(x_0 C_m)\Big)^\omega = \Pr_{A_{x_0}}(x_0 C_m^\omega) = \Pr_{A_{x_0}}(C_m^\omega)$. Using Lemma 6.33 we obtain $C_m^\omega = \{\rho \in X^\omega \mid \inf(\rho) = X\}$. This completes the proof of (6.13).

Therefore, we obtain that $E_p^Q = \varepsilon + \sum_{q \in Q} E_{p,q}^Q + \sum_{\emptyset \neq R \subseteq Q} E^Q_{p,\inf=R}$ is a probabilistic regular expression. Using Lemma 6.18, we restrict this expression to the valid summands:

$$E_p = \mathbb{1}_F(p)\,\varepsilon + \sum_{q \in F} E_{p,q}^Q + \sum_{R \in \mathcal{R}} E^Q_{p,\inf=R}.$$

By (6.12) and (6.13) we obtain

$$\|A\| = \sum_{q \in Q} \mu(q)\,\|E_q\|.$$

This completes the proof, as $E = \sum_{q \in Q} \mu(q)\, E_q$ is the desired probabilistic regular expression with $\|A\| = \|E\|$. ∎

Using the results of Sections 6.4 and 6.5 we have now shown the following theorem:

Theorem 6.36. Let $S \colon \Sigma^\infty \to [0,1]$ be a function. The following statements are equivalent:

1. $S = \|A\|$ for a probabilistic Muller-automaton A.
2. $S = \|E\|$ for a probabilistic regular expression E.

Moreover, the translations between probabilistic Muller-automata and probabilistic regular expressions are effective.

Chapter 7

Probabilistic Regular Expressions on Finite Trees

We will extend the notion of probabilistic regular expressions, which we developed for words in the last chapter, to finite ranked trees in this chapter.

This chapter is structured as the last chapter: in Section 7.1 we recall the notion of regular tree expressions for classical tree languages. Afterwards, in Section 7.2, we introduce probabilistic versions of the classical regular operations. Using these definitions we define probabilistic regular tree expressions in Section 7.3. Finally, we use Sections 7.4 and 7.5 to show the expressive equivalence of probabilistic regular tree expressions and probabilistic tree automata.

The results of this chapter have been published in [W15].

7.1 Regular Tree Expressions

Before we define probabilistic regular tree expressions, we recall the notion of regular tree expressions. Regular tree expressions play the same role to recognizable tree languages as regular expressions play to recognizable word languages. In contrast to regular expressions on words, regular tree expressions make use of an additional finite set of *variables*. This is necessary to mark the leaf nodes in a tree at which substitutions can occur. In the word case, concatenation always appends to every word in a language. Let V be a finite set of variables, we write $\mathrm{T}_\Sigma(V)$ for all trees over the rank alphabet Σ', where $\Sigma'_n = \Sigma_n$ for $n \geq 1$ and $\Sigma'_0 = \Sigma_0 \cup V$, i.e., $\mathrm{T}_\Sigma(V) = \mathcal{T}_{\mathbb{Q}}\Sigma'$.

Definition 7.1. Let $L, K \subseteq \mathrm{T}_\Sigma(V)$, $t \in \mathrm{T}_\Sigma(V)$ and $z \in V$. We make the following definitions:

1. The *tree concatenation* $t \cdot_z K \subseteq \mathrm{T}_\Sigma(V)$ of a tree $t \in \mathrm{T}_\Sigma(V)$ and a tree language

$K \subseteq \mathrm{T}_\Sigma(V)$ is given inductively by

$$a \cdot_z K = \begin{cases} \{a\} & \text{if } a \neq z \\ K & \text{if } a = z \end{cases}$$
$$f(t_1, \ldots, t_n) \cdot_z K = \big\{ f(s_1, \ldots, s_n) \;\big|\; s_i \in t_i \cdot_z K \text{ for } 1 \leq i \leq n \big\},$$

for all $a \in \Sigma_0 \cup V$ and $t = f(t_1, \ldots, t_n)$ with $n \geq 1$.

2. The *tree concatenation* $L \cdot_z K \subseteq \mathrm{T}_\Sigma(V)$ of two tree languages $L, K \subseteq \mathrm{T}_\Sigma(V)$ is

$$L \cdot_z K = \bigcup_{t \in L} t \cdot_z K.$$

3. The *Kleene iteration* $L^{*z} \subseteq \mathrm{T}_\Sigma(V)$ of a tree language $L \subseteq \mathrm{T}_\Sigma(V)$ is defined by

$$L^{*z} = \bigcup_{n \geq 0} L^{n,z} \quad \text{where} \quad L^{0,z} = \{z\} \quad \text{and} \quad L^{n+1,z} = L \cdot_z L^{n,z} \cup L^{n,z}.$$

Definition 7.2. The set of all *regular tree expressions RTE* is given in BNF by

$$E ::= \emptyset \mid z \mid f(\underbrace{E, \ldots, E}_{arity(f)\text{-times}}) \mid E \cup E \mid E \cdot_z E \mid E^{*z}.$$

The *language* $\mathrm{L}(E)$ of a regular tree expression E is the tree language inductively defined by

$$\begin{aligned}
\mathrm{L}(\emptyset) &= \emptyset \\
\mathrm{L}(z) &= \{z\} \\
\mathrm{L}(f(E_1, \ldots, E_n)) &= \big\{ f(s_1, \ldots, s_n) \;\big|\; s_i \in \mathrm{L}(E_i) \text{ for } 1 \leq i \leq n \big\} \\
\mathrm{L}(E_1 \cup E_2) &= \mathrm{L}(E_1) \cup \mathrm{L}(E_2) \\
\mathrm{L}(E_1 \cdot_z E_2) &= \mathrm{L}(E_1) \cdot_z \mathrm{L}(E_2) \\
\mathrm{L}(E^{*z}) &= \mathrm{L}(E)^{*z}.
\end{aligned}$$

A tree language L is called *regular* or *rational* if there is a regular tree expression E over some set of variables with $\mathrm{L}(E) = L$.

Example 7.3. Let $\Sigma = \{\mathrm{f}, \mathrm{a}, \mathrm{b}\}$, where f is a binary symbol and a, b are leaf symbols. Consider the following expression E:

$$E = \Big(\big(\mathrm{f}(\mathrm{y}, \mathrm{z}) \cup \mathrm{f}(\mathrm{z}, \mathrm{y}) \big)^{*\mathrm{y}} \cdot_{\mathrm{y}} \mathrm{a} \Big) \cdot_{\mathrm{z}} \Big(\big(\mathrm{f}(\mathrm{z}, \mathrm{z}) \big)^{*\mathrm{z}} \cdot_{\mathrm{z}} (\mathrm{a} \cup \mathrm{b}) \Big).$$

We claim that $\mathrm{L}(E)$ contains all trees with at least one a labelled node. This can be seen as follows: the expression $\big(\mathsf{f}(\mathsf{y},\mathsf{z}) \cup \mathsf{f}(\mathsf{z},\mathsf{y})\big)^{*_\mathsf{y}}$ generates all trees of the form $g_1(g_2(\cdots g_n(\mathsf{y})\cdots))$ where $g_i(t)$ is either $\mathsf{f}(t,\mathsf{z})$ or $\mathsf{f}(\mathsf{z},t)$. So, intuitively, it generates one path to a leaf node by making left/right choices. Afterwards, the single y leaf is replaced by an a. Finally, the remaining labels z are substituted by arbitrary trees, as $\mathrm{L}\big(\big(\mathsf{f}(\mathsf{z},\mathsf{z})\big)^{*_\mathsf{z}} \cdot_\mathsf{z} (\mathsf{a} \cup \mathsf{b})\big) = \mathrm{T}_\Sigma$.

The following result is the analogon of Kleene's theorem for finite ranked trees is due to Thatcher and Wright [TW68].

Theorem 7.4. Let Σ be a rank alphabet and $L \subseteq \mathrm{T}_\Sigma$. The following statements are equivalent:

1. $L = \mathrm{L}(A)$ for a tree automaton A.
2. $L = \mathrm{L}(E)$ for a regular tree expression E.

7.2 Probabilistic Operations on Tree Series

In this section, we introduce probabilistic tree concatenation, which is defined as weighted tree concatenation introduced by Droste, Pech and Vogler [DPV05], but restricted to so-called substitution summable tree series. Afterwards, we give a new iteration operation, the infinity-iteration, which will replace Kleene-iteration in probabilistic regular tree expressions.

We call any function $S\colon \mathrm{T}_\Sigma(V) \to [0,1]$ a *probabilistic tree series* or just a *tree series*. In order to ease the notation in the rest of the chapter, we introduce the substitution order. Intuitively, $s \trianglelefteq_W t$ holds if s can be obtained from t by removing some subtrees of t and inserting elements from $W \subseteq V$ in their place.

Definition 7.5. Let $W \subseteq V$. We define the *substitution order* $\trianglelefteq_W$ on $\mathrm{T}_\Sigma(V)$ by

$$s \trianglelefteq_W t \iff \mathrm{pos}(s) \subseteq \mathrm{pos}(t) \text{ and } s(x) = t(x) \text{ for all } x \in \mathrm{pos}(s) \setminus \mathrm{pos}_W(s).$$

Let $z \in V$. For convenience, we write $\trianglelefteq_z$ instead of $\trianglelefteq_{\{z\}}$.

The following restriction will ensure the well-definedness of probabilistic tree concatenation and infinity iteration. We say a tree series S is substitution summable, if, intuitively, it is for all trees t (at most) a distribution on the trees s which can be extended to t by substituting any variable from V.

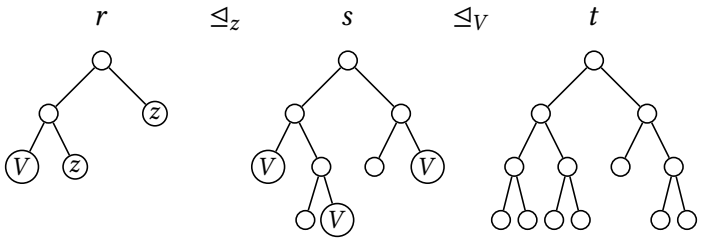


Figure 7.1: Situation in proof of Lemma 7.8

Definition 7.6. A tree series S is called *substitution summable* if

$$\sum_{s \trianglelefteq_V t} S(s) \leq 1,$$

for all $t \in \mathrm{T}_\Sigma(V)$.

We define probabilistic tree concatenation using the same expression as weighted tree concatenation over the semiring of non-negative real numbers, but restrict the operands to substitution summable tree series.

Definition 7.7. Let S be a substitution summable tree series and T be a tree series. We define the *tree concatenation* $S \cdot_z T$ of S and T by

$$(S \cdot_z T)(t) = \sum_{s \trianglelefteq_z t} S(s) \prod_{x \in \mathrm{pos}_z(s)} T(t|_x). \tag{7.1}$$

Next, we show that our definition is sound, i.e., the tree series $S \cdot_z T$ is well-defined and that $\cdot_z$ preserves substitution summability.

Lemma 7.8. Let S and T be tree series and S be substitution summable. The following statements hold:

1. $S \cdot_z T$ is again a probabilistic tree series, i.e., it only attains values in $[0, 1]$.
2. If T is also substitution summable, so is $S \cdot_z T$.

Proof. **1.** As S and T only map to positive values, it is clear, that $(S \cdot_z T)(t) \geq 0$ for all $t \in \mathrm{T}_\Sigma(V)$. Consider an arbitrary tree t. We obtain

$$(S \cdot_z T)(t) = \sum_{s \trianglelefteq_z t} S(s) \prod_{x\, \mathrm{pos}_z(s)} T(t|_x) \leq \sum_{s \trianglelefteq_z t} S(s) \leq \sum_{s \trianglelefteq_V t} S(s) \leq 1.$$

Thus, $S \cdot_z T$ is a probabilistic tree series.

2. Now, assume that T is substitution summable. Let $t \in \mathrm{T}_\Sigma(V)$. We have

$$\sum_{s \trianglelefteq_V t} (S \cdot_z T)(s) = \sum_{s \trianglelefteq_V t} \sum_{r \trianglelefteq_z s} S(r) \prod_{x \in \mathrm{pos}_z(r)} T(s|_x). \tag{7.2}$$

In order to use the assumptions on S and T we apply the following index transformation to the double sum: let $X = \{(r,s) \mid r \trianglelefteq_z s \trianglelefteq_V t\}$ and $Y = \{(r, (s_x)_{x \in \mathrm{pos}_z(r)}) \mid r \trianglelefteq_V t,\ s_x \trianglelefteq_V t|_x \text{ for all } x \in \mathrm{pos}_z(r)\}$. We show that the mapping $g \colon X \to Y$ given by $g(r,s) = (r, (s|_x)_{x \in \mathrm{pos}_z(r)})$ is bijective. Let $(r, (s_x)_{x \in \mathrm{pos}_z(r)}) \in Y$. We construct a tree s by replacing in r every occurrence x of z by s_x. More formally $s = r[x \leftarrow s_x]_{x \in \mathrm{pos}_z(r)}$. Clearly, we have $r \trianglelefteq_z s$. Moreover, we obtain $\mathrm{pos}(s) = \mathrm{pos}(r) \cup \bigcup_{x \in \mathrm{pos}_z(r)} x\,\mathrm{pos}(s_x) \subseteq \mathrm{pos}(t)$, as $\mathrm{pos}(r) \subseteq \mathrm{pos}(t)$ and $x\,\mathrm{pos}(s_x) \subseteq x\,\mathrm{pos}(t|_x) \subseteq \mathrm{pos}(t)$. In the same way, we obtain that s and t agree on $\mathrm{pos}(s) \setminus \mathrm{pos}_V(s)$ as $r \trianglelefteq_V t$ and $s_x \trianglelefteq_V t|_x$. Thus, g is surjective.

Next, assume $g(r,s) = g(r',s')$ for some $(r,s), (r',s') \in X$. By definition of g we get $r = r'$. Let $x \in \mathrm{pos}(s) \cap \mathrm{pos}(s')$. If $x \in \mathrm{pos}(r) \setminus \mathrm{pos}_z(r)$, then $s(x) = r(x) = r'(x) = s'(x)$, as $r \trianglelefteq_z s$ and $r' \trianglelefteq_z s'$. Assume $x \in (\mathrm{pos}(s) \setminus \mathrm{pos}(r)) \cup \mathrm{pos}_z(r)$. Let $x' \preceq x$ maximal with $x' \in \mathrm{pos}(r)$. Then $r(x') = z$, since either $x' = x \in \mathrm{pos}_z(r)$ or $x \in \mathrm{pos}(s) \setminus \mathrm{pos}(r)$ and x' is labelled by a leaf symbol in r but not in s. Moreover, r and s are only allowed to differ on $\mathrm{pos}_z(r)$. Let $x = x'x''$. We obtain $s(x) = s_{x'}(x'') = s'_{x'}(x'') = s'(x)$. So, s and s' coincide on $\mathrm{pos}(s) \cap \mathrm{pos}(s')$. As we are dealing with ranked trees, this implies $s = s'$. Therefore, g is bijective.

We continue (7.2):

$$\begin{aligned}\sum_{s \trianglelefteq_V t} (S \cdot_z T)(s) &= \sum_{s \trianglelefteq_V t} \sum_{r \trianglelefteq_z s} S(r) \prod_{x \in \mathrm{pos}_z(r)} T(s|_x) = \sum_{r \trianglelefteq_V t} \sum_{\substack{s_x \trianglelefteq_V t|_x \\ (x \in \mathrm{pos}_z(r))}} S(r) \prod_{x \in \mathrm{pos}_z(r)} T(s_x) \\ &= \sum_{r \trianglelefteq_V t} S(r) \prod_{x \in \mathrm{pos}_z(r)} \sum_{s_x \trianglelefteq_V t|_x} T(s_x) \leq \sum_{r \trianglelefteq_V t} S(r) \leq 1.\end{aligned}$$

This shows that $S \cdot_z T$ is substitution summable. ∎

As probabilistic tree concatenation is just weighted tree concatenation restricted to substitution summable tree series, associativity directly carries over.

Lemma 7.9. Let $R, S, T \colon \mathrm{T}_\Sigma(V) \to [0,1]$ be probabilistic tree series and $z \in V$. Then

$$R \cdot_z (S \cdot_z T) = (R \cdot_z S) \cdot_z T.$$

This equality does not hold in general, if two distinct variables are used in the products.

Proof. Use distributivity and an index transformation similar to the one in the proof of Lemma 7.8. For details, see [DPV05]. ∎

We define the powers of a tree series S with respect to a variable z:

$$S^{0,z} = \mathbb{1}_{\{z\}} \qquad \text{and} \qquad S^{n+1,z} = S^{n,z} \cdot_z S.$$

We will make use of the following representation of $S^{n,z}$ which can be obtained using distributivity:

$$S^{n,z}(t) = \sum_{t_1 \trianglelefteq_z t_2 \trianglelefteq_z \cdots \trianglelefteq_z t_{n-1} \trianglelefteq_z t_n = t} S(t_1) \prod_{i=1}^{n-1} \prod_{x \in \mathrm{pos}_z(t_i)} S(t_{i+1}|_x).$$

Next, we give the definition of infinity iteration. This will be the iteration operation that we will use in probabilistic regular tree expressions. There is a conceptional difference to standard Kleene-iteration: in Kleene-iteration, there is a choice after substituting a variable by a tree to either continue the process and substitute the variables in that tree or to stop. In infinity iteration this choice is removed, variables have to be substituted for as long as possible.

Definition 7.10. Let S be a substitution summable tree series and $z \in V$. We define the *infinity iteration* $S^{\infty z}$ of S by

$$S^{\infty z}(t) = \lim_{n \to \infty} S^{n,z}(t),$$

for all trees t.

One advantage of using infinity iteration is that it works well with substitution summable tree series: the infinity iteration of a substitution summable tree series is always well-defined, bounded by 1, and is itself substitution summable.

Lemma 7.11. Let S be a substitution summable probabilistic tree series and $z \in V$. The following statements hold:

1. $S^{\infty z}$ is well-defined, i.e., $S^{\infty z}(t) \in [0, 1]$ for all $t \in \mathrm{T}_\Sigma(V)$.
2. $S^{\infty z}$ is again substitution summable.
3. $S^{\infty z}(t) = 0$ if $\mathrm{pos}_z(t) \neq \emptyset$ and $S(z) < 1$.

Proof. If $S(z) = 1$, then $S = \mathbb{1}_{\{z\}}$ as S is substitution summable. Hence, $S^{\infty z} = S$ is well-defined and substitution summable. We show statements 1 - 3 for the case $S(z) < 1$. We start with statement 3, which is needed for the first statement.

3. We show that $S^{\infty z}(t) = 0$ if $\mathrm{pos}_z(t) \neq \emptyset$. Note that, since $\mathrm{pos}_z(t) \neq \emptyset$, we also have $\mathrm{pos}_z(s) \neq \emptyset$ for all $s \trianglelefteq_z t$. Let $\mathcal{C}_k^n(t)$ contain all $n+1$-tuples $(t_1, \ldots, t_{n+1})$ of trees with $t_i \trianglelefteq_z t_{i+1}$ for all $1 \leq i \leq n$, and $t_{n+1} = t$, such that there are exactly k indices $j \in \{1, \ldots, n\}$ with $t_j = t_{j+1}$. Using that $(\mathcal{C}_k^n(t))_{k=0}^n$ is a partition of all chains of length $n+1$ below t and that at least one substitution of z by itself has to occur if $t_j = t_{j+1}$ we obtain

$$S^{n+1,z}(t) = \sum_{t_1 \trianglelefteq_z \cdots \trianglelefteq_z t_{n+1} = t} S(t_1) \prod_{i=1}^{n} \prod_{x \in \mathrm{pos}_z(t_i)} S(t_{i+1}|_x) \leq \sum_{k=0}^{n} \sum_{(t_1,\ldots,t_{n+1}) \in \mathcal{C}_k^n(t)} S(z)^k.$$

Hence, we need to find an upper bound for $|\mathcal{C}_k^n(t)|$. Let $N(t)$ be the number of all trees s with $s \trianglelefteq_z t$. Note that, by definition of $\mathcal{C}_k^n(t)$, there are $n+1-k$ distinct trees in every tuple in $\mathcal{C}_k^n(t)$. Thus, $\mathcal{C}_k^n(t) = \emptyset$ if $n+1-k > N(t)$, i.e., $k < n+1-N(t)$. The mapping $g \colon \mathcal{C}_k^n(t) \to \mathcal{P}(\{1, \ldots, n\} \times \{s \mid s \trianglelefteq_z t\})$ defined by $g(t_1, \ldots, t_n) = \{(i, t_i) \mid 1 \leq i \leq n,\ t_i \neq t_{i+1}\}$ is injective and $|g(t_1, \ldots, t_{n+1})| = n - k$ for every tuple $(t_1, \ldots, t_{n+1}) \in \mathcal{C}_k^n(t)$. We can consider $g(t_1, \ldots, t_{n+1})$ as partial function on $\{1, \ldots, n\}$. Moreover, $g(t_1, \ldots, t_{n+1})$ is strictly monotonic. Therefore,

$$|\mathcal{C}_k^n(t)| = |g(\mathcal{C}_k^n(t))| \leq \binom{n}{n-k}\binom{N(t)-1}{n-k} = \binom{n}{k}\binom{N(t)-1}{n-k}.$$

The additional "-1" is used as we did not include $t_{n+1} = t$ in $g(t_1, \ldots, t_{n+1})$. Thus, we obtain the following:

$$\begin{aligned}
S^{n,z}(t) &\leq \sum_{k=0}^{n-1} |\mathcal{C}_k^n(t)|\, S(z)^k && = \sum_{k=0}^{n} \binom{n}{k}\binom{N(t)-1}{n-k} S(z)^k \\
&= \sum_{k=n+1-N(t)}^{n} \binom{n}{k}\binom{N(t)-1}{n-k} S(z)^k && \leq S(z)^{n+1-N(t)} \sum_{k=n+1-N(t)}^{n} \binom{n}{k}\binom{N(t)-1}{n-k} \\
&\leq S(z)^n \cdot \underbrace{S(z)^{1-N(t)} N(t)! \sum_{k=0}^{N(t)-1} \cdot \binom{n}{k}}_{=P(n)}
\end{aligned}$$

The polynomial $P(n)$ has degree $N(t)$ independent of n. Thus, $S^{n,z}(t) \to 0$ as $n \to \infty$ since $S(z) < 1$. Hence, $S^{\infty z}(t) = 0$ as claimed.

1. We consider the case $\mathrm{pos}_z(t) = \emptyset$ and show $S^{\infty z}(t) \in [0,1]$. For this, we prove the following monotonicity property $S^{n,z}(t) \leq S^{n+1,z}(t)$:

$$S^{n,z}(t) = \sum_{t_1 \trianglelefteq_z \cdots \trianglelefteq_z t_{n+1} = t} S(t_1) \prod_{i=1}^{n} \prod_{x \in \mathrm{pos}_z(t_i)} S(t_{i+1}|_x)$$

As $\mathrm{pos}_z(t) = \emptyset$ we just add an empty product in the next step:

$$= \sum_{t_1 \trianglelefteq_z \cdots \trianglelefteq_z t_{n+1} = t_{n+2} = t} S(t_1) \prod_{i=1}^{n+1} \prod_{x \in \mathrm{pos}_z(t_i)} S(t_{i+1}|_x)$$

$$\leq \sum_{t_1 \trianglelefteq_z \cdots \trianglelefteq_z t_{n+1} \trianglelefteq_z t_{n+2} = t} S(t_1) \prod_{i=1}^{n+1} \prod_{x \in \mathrm{pos}_z(t_i)} S(t_{i+1}|_x) = S^{n+1,z}(t).$$

As $S^{n,z}(t) \leq 1$ for every tree t by Lemma 7.8, the sequence $(S^{n,z}(t))_{n \geq 1}$ is monotonically increasing and bounded. Thus, the sequence converges and the limit also bounded by 1.

2. We show that $S^{\infty z}$ is substitution summable. Let $t \in \mathrm{T}_\Sigma(V)$. We conclude

$$\sum_{s \trianglelefteq_V t} S^{\infty z}(s) = \sum_{s \trianglelefteq_V t} \lim_{n \to \infty} S^{n,z}(s) = \lim_{n \to \infty} \sum_{s \trianglelefteq_V t} S^{n,z}(s) \leq 1,$$

where we could interchange sum and limit, as the sum has a finite index set. Moreover, we have $\sum_{s \trianglelefteq_V} S^{n,z}(s) \leq 1$ by Lemma 7.8. ∎

Instead of defining $S^{\infty z}$ as limit of powers of S as in Definition 7.10, one can also characterise $S^{\infty z}$ as unique solution of an equation.

Lemma 7.12. Let $z \in V$ and S be a substitution summable tree series with $S(z) < 1$. Then, $S^{\infty z}$ is the unique solution of the equation $X = S \cdot_z X$.

Proof. We first show, that $S^{\infty z}$ is a solution of $X = S \cdot_z X$. We have the following:

$$(S \cdot_z S^{\infty z})(t) = \sum_{s \trianglelefteq_z t} S(s) \prod_{x \in \mathrm{pos}_z(s)} \lim_{n \to \infty} S^{n,z}(t|_x)$$

As the product and sum are finite, these can be interchanged with the limit:

$$= \lim_{n \to \infty} \sum_{s \trianglelefteq_z t} S(s) \prod_{x \in \mathrm{pos}_z(s)} S^{n,z}(t|_x)$$

$$= \lim_{n \to \infty} (S \cdot_z S^{n,z})(t) = \lim_{n \to \infty} S^{n+1,z}(t) = S^{\infty,z}(t).$$

Thus, $S^{\infty z}$ is a solution of $X = S \cdot_z X$.

Next, we show that every solution is equal to $S^{\infty z}$. Let T be any substitution summable probabilistic tree series with $T = S \cdot_z T$. Thus, $T = S^{n,z} \cdot_z T$ for every $n \geq 0$. Taking n to the limit to infinity, we obtain:

$$T(t) = \lim_{n\to\infty} T(t) = \lim_{n\to\infty} (S^{n,z} \cdot_z T)(t)$$

As before, we can interchange the limit with the finite sum:

$$\begin{aligned} &= \sum_{s \trianglelefteq_z t} \left(\lim_{n\to\infty} S^{n,z}(s) \right) \prod_{x \in \mathrm{pos}_z(s)} T(t|_x) \\ &= \sum_{s \trianglelefteq_z t} S^{\infty z}(s) \prod_{x \in \mathrm{pos}_z(s)} T(t|_x) \end{aligned}$$

By Lemma 7.11, we have $S^{\infty z}(s) = 0$ if $\mathrm{pos}_z(s) \neq \emptyset$. Moreover, $s \trianglelefteq_z t$ with $s \neq t$ implies $\mathrm{pos}_z(s) \neq \emptyset$. Thus

$$\begin{aligned} &= \begin{cases} S^{\infty z}(t) & \text{if } \mathrm{pos}_z(t) = \emptyset \\ 0 & \text{otherwise} \end{cases} \\ &= S^{\infty z}(t). \end{aligned}$$

This completes the proof that $S^{\infty z}$ is the unique solution of $X = S \cdot_z X$. ∎

In [DPV05] another quantitative iteration for weighted tree series was proposed: a weighted Kleene-iteration. We first restate their definition and then show how weighted Kleene-iteration relates to probabilistic infinite-iteration. Though the weighted definitions work with arbitrary semirings, we will only use the semiring of the positive real numbers here. The product of arbitrary functions $S, T \colon \mathrm{T}_\Sigma(V) \to [0, \infty)$ is also defined by the right side of (7.1).

Definition 7.13. Let $S \colon \mathrm{T}_\Sigma(V) \to [0, \infty)$ be any function with $S(z) = 0$. We define functions $S_z^{n,\mathrm{F}}$ for $n \geq 0$ by

$$S_z^{0,\mathrm{F}} = \mathbb{0} \qquad \text{and} \qquad S_z^{n+1,\mathrm{F}} = S \cdot_z (S_z^{n,\mathrm{F}} + \mathbb{1}_{\{z\}}),$$

where $\mathbb{0}$ is the null-function. Using these definitions, we set $S_z^{*,\mathrm{F}}(t) = S_z^{\mathrm{height}(t),\mathrm{F}}$.

Intuitively, if a tree t does not contain the variable z, then the computation of $S_z^{*,\mathrm{F}}$ has to continue, i.e., the term $\mathbb{1}_{\{z\}}$ is always zero, until no more z's occur. Thus, the value of $S_z^{*,\mathrm{F}}(t)$ equals $S^{\infty z}(t)$. This is the statement of the next lemma.

Lemma 7.14. Let S be a substitution summable probabilistic tree series and $z \in V$ with $S(z) = 0$. Then, $S^{\infty z}(t) = S_z^{*,\mathrm{F}}(t)$ for all $t \in \mathrm{T}_\Sigma(V \setminus \{z\})$.

Proof. We show $S^{n,z}(t) = S_z^{n,\mathrm{F}}(t)$ for all $t \in \mathrm{T}_\Sigma(V \setminus \{z\})$. Let $t \in \mathrm{T}_\Sigma(V \setminus \{z\})$. For $n = 0$, we have $S^{0,z}(t) = \mathbb{1}_{\{z\}}(t) = 0 = \mathbb{0}(t) = S_z^{0,\mathrm{F}}(t)$. Next, assume the statement holds for n. We obtain

$$S_z^{n+1,\mathrm{F}}(t) = \sum_{s \trianglelefteq_z t} S(s) \prod_{x \in \mathrm{pos}_z(s)} \left(S_z^{n,\mathrm{F}}(t|_x) + \mathbb{1}_{\{z\}}(t|_x) \right)$$

By assumption, t does not contain the label z. Thus, $\mathbb{1}_{\{z\}}(t|_x)$ is always zero:

$$= \sum_{s \trianglelefteq_z t} S(s) \prod_{x \in \mathrm{pos}_z(s)} S^{n,z}(t)$$
$$= S^{n+1,z}(t).$$

In [DPV05] it was shown that $S_z^{n,\mathrm{F}}(t) = S_z^{\mathrm{height}(t),\mathrm{F}}(t)$ for all $n \geq \mathrm{height}(t)$. Thus, we conclude

$$S^{\infty z}(t) = \lim_{n \to \infty} S^{n,z}(t) = \lim_{n \to \infty} S_z^{n,\mathrm{F}}(t) = S_z^{\mathrm{height}(t),\mathrm{F}}(t) = S_z^{*,\mathrm{F}}(t).$$

This completes the proof. ∎

7.3 Syntax and Semantics of Probabilistic Regular Tree Expressions

In this section, we define the syntax and semantics of probabilistic regular tree expressions and give an example of these definitions at work.

Definition 7.15. The set PRTE of all *probabilistic regular tree expressions* is the smallest set R satisfying the following properties:

1. $\mathbb{0} \in R$
2. $z \in R$ for every $z \in V$
3. $\sum_{f \in \Sigma'} f(E_1^{(f)}, \ldots, E_{arity(f)}^{(f)}) \in R$ for all $\Sigma' \subseteq \Sigma$ and families of expressions $\left(E_i^{(f)}\right)_{f \in \Sigma, i \leq arity(f)}$ in R
4. $pE + (1-p)F \in R$ for all $E, F \in R$ and $p \in [0,1]$
5. $E \cdot_z F \in R$ for all $E, F \in R$ and $z \in V$

6. $E^{\infty z} \in R$ for all $E \in R$ and $z \in V$

As in the word case, the restricted sums do not allow for the usual associativity, commutativity, and distributivity laws to hold any longer. Thus, these are explicitly added as additional identities to R:

7. $E \cdot_z (F \cdot_z G) \equiv (E \cdot_z F) \cdot_z G$, $E + (F + G) \equiv (E + F) + F$, and $p_1(p_2 E) \equiv (p_1 p_2)E$

8. $E + F \equiv F + E$

9. $(E + F) \cdot_z G \equiv E \cdot_z G + F \cdot_z G$, $p(E + F) \equiv pE + pF$, and $(p_1 + p_2)E \equiv p_1 E + p_2 E$

Recall that each identity states that an expression containing the left side of an identity as a subexpression is in R if and only if the same expression, but with this subexpression replaced by the right side of the identity, is in R and vice versa.

The semantics of a probabilistic regular tree expressions is defined using structural induction on the syntax tree:

$$
\begin{aligned}
\|\mathbb{0}\|(t) &= 0, \\
\|z\|(t) &= \begin{cases} 1 & \text{if } t = z \\ 0 & \text{otherwise,} \end{cases} \\
\|f(E_1, \ldots, E_n)\|(t) &= \begin{cases} \prod_{i=1}^{n} \|E_i\|(t_i) & \text{if } f = g \\ 0 & \text{otherwise,} \end{cases} \\
\|pE\|(t) &= p\|E\|(t), \\
\|E + F\|(t) &= \|E\|(t) + \|F\|(t), \\
\|E \cdot_z F\|(t) &= \big(\|E\| \cdot_z \|F\|\big)(t), \\
\|E^{\infty z}\|(t) &= \|E\|^{\infty z}(t),
\end{aligned}
$$

for all $t = g(t_1, \ldots, t_n) \in \mathrm{T}_\Sigma(V)$ with $n \geq 0$.

The following lemma is a direct consequence of the results in the last section.

Lemma 7.16. Let E be a probabilistic regular tree expression. Then $\|E\|$ is a well-defined function $\|E\| \colon \mathrm{T}_\Sigma(V) \to [0, 1]$. Moreover, $\|E\|$ is substitution summable.

Proof. Let M contain all PRTEs which satisfy the statement of the lemma. Clearly, the null-function and $\mathbb{1}_z$ are well-defined and substitution summable. Thus, $0 \in M$ and $z \in M$.

Assume $E_i^{(g)} \in M$ for all $g \in \Sigma$ and $i \leq \mathit{arity}(g)$ and $E = \sum_{g\in\Sigma} g(E_1^{(g)}, \ldots, E_{\mathit{arity}(g)}^{(g)})$. Let $t = f(t_1, \ldots, t_n)$. Hence, $\|E\| = \prod_{i=1}^{\mathit{arity}(f)} \|E_i^{(f)}\|(t_i) \leq 1$ by induction hypothesis. We show that $\|E\|$ is substitution summable:

$$\sum_{s \trianglelefteq_V t} \|E\|(s) = \|E\|(z) + \sum_{\substack{s_i \trianglelefteq_z t_i \\ (i=1,\ldots,\mathit{arity}(f))}} \prod_{i=1}^{\mathit{arity}(f)} \|E_i^{(f)}\|(s_i) = \prod_{i=1}^{\mathit{arity}(f)} \sum_{s \trianglelefteq_z t_i} \|E_i^{(f)}\|(s) \leq 1,$$

since $\|E\|(z) = 0$ as only elements of Σ are included in the definition of E. Hence, $E \in M$.

Finally, if $E, F \in M$, we obtain $E \cdot_z F \in M$ by Lemma 7.8, and $E^{\infty z} \in M$ by Lemma 7.11. Furthermore, application of the ACD rules does not change the semantics and so, membership in M. Thus, M satisfies all closure properties of Definition 7.15. Therefore, $M = \mathrm{PRTE}$. ∎

At the end of this section, we give an example how probabilistic regular tree expressions can be used to define probabilistic tree series.

Example 7.17. We come back to Example 7.3. The regular tree expression given there is not a probabilistic regular tree expression for two reasons: first, it contains the Kleene-iteration and not the infinity-iteration, and second, the sum f(y, z)+f(z, y) is not allowed in PRTE. In fact, we will later see, that the characteristic function of the language described by the expression given in Example 7.3 cannot be recognized by a probabilistic top-down tree automaton.

In order to give a probabilistic variant of this expression, we replace this sum by a probabilistic choice: let $\Sigma = \{\mathsf{f}, \mathsf{a}, \mathsf{b}\}$ we define the PRTE E by

$$E = \underbrace{\left(\frac{1}{2}\,\mathsf{f}(\mathsf{y},\mathsf{z}) + \frac{1}{2}\,\mathsf{f}(\mathsf{z},\mathsf{y}) + \mathsf{a}\right)^{\infty\mathsf{y}}}_{=E_1} \cdot_{\mathsf{z}} \underbrace{\left(\mathsf{f}(\mathsf{z},\mathsf{z}) + \mathsf{a} + \mathsf{b}\right)^{\infty\mathsf{z}}}_{=E_2}.$$

We obtained the expression $1/2\,\mathsf{f}(\mathsf{y},\mathsf{z}) + 1/2\,\mathsf{f}(\mathsf{z},\mathsf{y}) + \mathsf{a}$ using distributivity from the expression $1/2(\mathsf{f}(\mathsf{y},\mathsf{z}) + \mathsf{a}) + 1/2(\mathsf{f}(\mathsf{z},\mathsf{y}) + \mathsf{a})$, which in turn can be directly constructed from the definition. The expression E_1 assigns the probability $(1/2)^n$ to all trees of the form $g_1(g_2(\cdots g_n(\mathsf{a})\cdots))$ where either $g_i(t) = \mathsf{f}(t, \mathsf{z})$ or $g_i(t) = \mathsf{f}(\mathsf{z}, t)$ for all $1 \leq i \leq n$. For all trees t not of this form, we have $\|E_1\|(t) = 0$. The second part of the expression, E_2, assigns probability 1 to every tree in T_Σ.

Given an arbitrary tree $t \in \mathrm{T}_\Sigma$ and a position $x \in \mathrm{pos}(t)$ with $t(x) = \mathsf{a}$, let s be the tree obtained from t replacing all subtrees which are not on the direct path to x by z's. Thus, we have $\|E_1\|(s) = (1/2)^{|x|}$. Conversely, every tree $s \trianglelefteq_{\mathsf{z}} t$ with

$\|E_1\|(s) > 0$ uniquely identifies a position x with $t(x) = \mathrm{a}$. Therefore, we obtain

$$\|E\|(t) = \sum_{s \trianglelefteq_z t} \|E_1\|(s) \prod_{x \in \mathrm{pos}_z(s)} \|E_2\|(t|_x) = \sum_{s \trianglelefteq_z t} \|E_1\|(s) = \sum_{x \in \mathrm{pos}_\mathrm{a}(t)} \left(\frac{1}{2}\right)^{|x|}.$$

7.4 From Expressions to Automata

In the next two sections, we show the expressive equivalence of probabilistic regular tree expressions and probabilistic tree automata. In this section, we give an inductive construction of a tree automaton for a given expression. In Section 6.5 we show the converse direction.

As shown in Lemma 7.16, every function $\|E\|$ for a given expression E is substitution summable. In the following definition, we give a syntactic restriction on a tree automaton A which ensures that $\|A\|$ is also substitution summable.

Definition 7.18. Let $A = (Q, \delta, \mu, F)$ be a probabilistic tree automaton over $\mathrm{T}_\Sigma(V)$. For $W \subseteq V \cup \Sigma_0$ let $F_W = \{q \in Q \mid (q, a) \in F \text{ for some } a \in W\}$. We say A is *substitution summable* if the $|V| + 1$ sets F_{Σ_0}, $F_{\{z\}}$ $(z \in V)$ are pairwise disjoint and the set F_V contains only sink states.

We will use the notation F_W throughout this chapter. For single variables $z \in V$, we write F_z for $F_{\{z\}}$.

Lemma 7.19. Let A be a substitution summable probabilistic tree automaton. Then $\|A\|$ is also substitution summable.

Proof. Let $A = (Q, \delta, \mu, F)$ and the sets F_W as in Definition 7.18. Instead of showing the actual statement $\sum_{s \trianglelefteq_V t} \|A\|(t) \leq 1$ for all $t \in \mathrm{T}_\Sigma(V)$, we prove a slightly stronger statement: let $\delta_q(t)$ be defined as below Definition 2.25. We show $\sum_{s \trianglelefteq_V t} \delta_q(s) \leq 1$ for all $q \in Q$ and $t \in \mathrm{T}_\Sigma(V)$ using induction on height(t) and showing the statement for all $q \in Q$ at the same time. Thus, let $q \in Q$ and $t \in \mathrm{T}_\Sigma(V)$. First, consider the case $t = a \in \Sigma_0$. We obtain

$$\sum_{s \trianglelefteq_V t} \delta_q(s) = \delta_q(a) + \sum_{z \in V} \delta_q(z) = \mathbb{1}_{F_{\{a\}}}(q) + \sum_{z \in V} \mathbb{1}_{F_{\{z\}}}(q) \overset{(\#)}{\leq} 1,$$

where (#) holds as the sets F_{Σ_0}, $F_{\{z\}}$ $(z \in V)$ are pairwise disjoint. The case $t = z \in V$ is analogous, only the term "$\delta_q(a)$" is left out.

Assume $t = f(t_1, \ldots, t_n)$. Note that a tree $s \trianglelefteq_V t$ is either of the form $s = z \in V$ or $s = f(s_1, \ldots, s_n)$ with $s_i \trianglelefteq_z t_i$ for all $i = 1, \ldots, n$. Thus, we have

$$\sum_{s \trianglelefteq_z t} \delta_q(s) = \sum_{z \in V} \delta_q(z) + \sum_{\substack{s_i \trianglelefteq_V t_i \\ (i=1,\ldots,n)}} \delta_q(f(s_1, \ldots, s_n))$$

$$= \sum_{z \in V} \mathbb{1}_{F_z}(q) + \sum_{\substack{s_i \trianglelefteq_V t_i \\ (i=1,\dots,n)}} \sum_{q_1,\dots,q_n \in Q} \delta(q,f)(q_1,\dots,q_n) \prod_{i=1}^{n} \delta_{q_i}(s_i)$$

As the sets F_z ($z \in V$) are pairwise disjoint, we have $\sum_{z \in V} \mathbb{1}_{F_z} = \mathbb{1}_{F_V}$.

$$= \mathbb{1}_{F_V}(q) + \sum_{q_1,\dots,q_n \in Q} \delta(q,f)(q_1,\dots,q_n) \prod_{i=1}^{n} \sum_{s \trianglelefteq_V t_i} \delta_{q_i}(s)$$

Applying the induction hypothesis to each of the δ_{q_i} we obtain

$$\leq \mathbb{1}_{F_V}(q) + \sum_{q_1,\dots,q_n \in Q} \delta(q,f)(q_1,\dots,q_n). \qquad (*)$$

If $q \in F_V$ holds, q is a sink state and $\delta(q,f) = \mathbb{0}$ Hence, $(*) = 1$. On the other hand, if $q \notin F_V$, we conclude $\mathbb{1}_{F_V}(q) = 0$ and thus $(*) \leq 1$, as $\delta(q,f)$ is a distribution on Q^n or $\mathbb{0}$. Finally, for any $t \in \mathrm{T}_\Sigma(V)$, we obtain

$$\sum_{s \trianglelefteq_V t} \|A\|(s) = \sum_{s \trianglelefteq_V t} \sum_{q \in Q} \mu(q)\delta_q(s) = \sum_{q \in Q} \mu(q) \sum_{s \trianglelefteq_V t} \delta_q(s) \leq 1.$$

This completes the proof. ∎

The rest of this section is devoted to showing that the class of tree series recognizable by substitution summable probabilistic tree automata satisfies the closure properties of Definition 7.15. Thus, we will obtain the result that every probabilistic regular tree expression is equivalent to a probabilistic tree automaton.

Lemma 7.20. Let $\Sigma' \subseteq \Sigma$ and $(A^i_f)_{f,i}$ for $f \in \Sigma'$, $1 \leq i \leq \mathit{arity}(f)$ be a family substitution summable probabilistic tree automata. There is a substitution summable probabilistic tree automaton A such that

$$\|A\|(t) = \begin{cases} \prod_{i=1}^{n} \|A^i_f\|(t_i) & \text{if } f \in \Sigma' \\ 0 & \text{otherwise} \end{cases}$$

for all $t = f(t_1,\dots,t_n) \in \mathrm{T}_\Sigma(V)$. Note that empty products are 1 by convention.

Proof. Let $A^i_f = (Q^i_f, \delta^i_f, \mu^i_f, F^i_f)$ for every $f \in \Sigma'$ and $1 \leq i \leq \mathit{arity}(f)$. We assume that the sets of states Q^i_f are pairwise disjoint. We define $A = (Q, \delta, \mu, F)$ where

$$Q = \{q_0\} \cup \bigcup_{\substack{f \in \Sigma' \\ 1 \leq i \leq \mathit{arity}(f)}} Q^i_f \quad \text{and} \quad F = \{(q_0, a) \mid a \in \Sigma' \cap \Sigma_0\} \cup \bigcup_{\substack{f \in \Sigma' \\ 1 \leq i \leq \mathit{arity}(f)}} F^i_f.$$

Furthermore, we let $\mu = \mathbb{1}_{\{q_0\}}$ and define δ by $\delta(q, f)$ for $q \in Q^i_{f'}$ by

$$\delta(q, f)(q_1, \ldots, q_n) = \begin{cases} \delta^i_{f'}(q_1, \ldots, q_n) & \text{if } q_j \in Q^i_{f'} \text{ for all } 1 \leq j \leq n \\ 0 & \text{otherwise} \end{cases}$$

and for $q = q_0$ by

$$\delta(q_0, f)(q_1, \ldots, q_n) = \begin{cases} \prod_{i=1}^n \mu^i_f(q_i) & \text{if } f \in \Sigma' \text{ and } q_i \in Q^i_f \text{ for all } 1 \leq i \leq n \\ 0 & \text{otherwise.} \end{cases}$$

Note that δ agrees with δ^i_f on Q^i_f. Thus, we also have $\delta_q = (\delta^i_f)_q$ for all $q \in Q^i_f$.

To show that A actually satisfies the statement of the lemma, first consider $t = a \in \Sigma_0$. We have $\|A\|(t) = \mathbb{1}_F(q_0, a) = \mathbb{1}_{\Sigma'}(a)$. For $t = f(t_1, \ldots, t_n)$ with $n \geq 1$ we have $\delta(q_0, f) = \mathbb{0}$ if $f \notin \Sigma'$ and thus also $\|A\|(t) = 0$. For $f \in \Sigma'$ we obtain

$$\begin{aligned} \|A\|(t) &= \sum_{q_1, \ldots, q_n \in Q} \delta(q_0, f)(q_1, \ldots, q_n) \prod_{i=1}^n \delta_{q_i}(t_i) \\ &= \sum_{q_1 \in Q^1_f, \ldots, q_n \in Q^n_f} \left(\prod_{i=1}^n \mu^i_f(q_i) \right) \prod_{i=1}^n \delta_{q_i}(t_i) \\ &= \prod_{i=1}^n \sum_{q \in Q^i_f} \mu^i_f(q) \, (\delta^i_f)_q(t_i) \\ &= \prod_{i=1}^n \|A^i_f\|(t_i). \end{aligned}$$

Thus, A has the claimed behaviour. It remains to show that A is substitution summable. Let $q \in F_V$. As $q_0 \notin F_V$, there is a $f \in \Sigma$ and $i \in \{1, \ldots, \mathit{arity}(f)\}$ with $q \in Q^i_f$ and thus $q \in (F^i_f)_V$. Hence, $\delta(q, f') = \delta^i_f(q, f') = \mathbb{0}$ as A^i_f is substitution summable. Next, assume $q \in F_{\{u\}} \cap F_{\{v\}}$ for some $u, v \in \Sigma_0 \cup V$ with $u \neq v$ and not both in Σ_0. As q_0 is not in any $F_{\{z\}}$, we have $q \neq q_0$. There is a $f \in \Sigma_0$ and $i \in \{1, \ldots, \mathit{arity}(f)\}$ with $q \in Q^i_f$ and therefore $q \in (F^i_f)_{\{u\}} \cap (F^i_f)_{\{v\}}$. A contradiction. This shows that A is substitution summable. ∎

Lemma 7.21. Let A_1 and A_2 be substitution summable probabilistic tree automata and $p \in [0, 1]$. There is a substitution summable probabilistic tree automaton A with $\|A\| = p\|A_1\| + (1 - p)\|A_2\|$.

Proof. Let $A_i = (Q_i, \delta_i, \mu_i, F_i)$ and assume that Q_1 and Q_2 are disjoint. We define $A = (Q, \delta, \mu, F)$ with $Q = Q_1 \cup Q_2$, $F = F_1 \cup F_2$,

$$\delta(q, f)(q_1, \dots, q_n) = \begin{cases} \delta_1(q, f)(q_1, \dots, q_n) & \text{if } q, q_1, \dots, q_n \in Q_1 \\ \delta_2(q, f)(q_1, \dots, q_n) & \text{if } q, q_1, \dots, q_n \in Q_2 \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\mu(q) = \begin{cases} p\,\mu_1(q) & \text{if } q \in Q_1 \\ (1-p)\,\mu_2(q) & \text{if } q \in Q_2. \end{cases}$$

Note that δ agrees with δ_1 on Q_2 and with δ_2 on q_2. For the behaviour of A we obtain

$$\begin{aligned} \|A\|(t) &= \sum_{q \in Q} \mu(q)\, \delta_q(t) = \sum_{q \in Q_1} p\,\mu_1(q)\, (\delta_1)_q(t) + \sum_{q \in Q_2} (1-p)\,\mu_2\, (\delta_2)_q(t) \\ &= p\,\|A_1\|(t) + (1-p)\,\|A_2\|(t). \end{aligned}$$

The substitution summability of A_1 and A_2 immediately carries over to A. ∎

Lemma 7.22. Let A_1 and A_2 be substitution summable probabilistic tree automata and $z \in V$. There is a substitution summable probabilistic tree automaton A with $\|A\| = \|A_1\| \cdot_z \|A_2\|$.

Proof. Let $A_i = (Q_i, \delta_i, \mu_i, F_i)$ for $i = 1, 2$. Let $X = (F_1)_z = \{q \in Q_1 \mid (q, z) \in F_1\}$. We define $A = (Q, \delta, \mu, F)$ by

$$Q = (Q_1 \setminus X) \cup Q_2, \qquad F = (F_1 \setminus (X \times \{z\})) \cup F_2,$$

$$\mu(q) = \begin{cases} \mu_1(q) & \text{if } q \in Q_1 \setminus X \\ \mu_1(X)\,\mu_2(q) & \text{if } q \in Q_2, \end{cases}$$

$$\delta(q, f)(q_1, \dots, q_n) = \begin{cases} \displaystyle\sum_{r_1, \dots, r_n \in Q_1} \delta_1(q, f)(r_1, \dots, r_n) \prod_{i=1}^{n} \kappa(r_i, q_i) & \text{if } q \in Q_1 \setminus X \\ \delta_2(q, f)(q_1, \dots, q_n) & \text{if } q, q_1, \dots, q_n \in Q_2 \\ 0 & \text{otherwise,} \end{cases}$$

where $\kappa \colon Q_1 \times Q \to [0, 1]$ is given by

$$\kappa(r, q) = \begin{cases} 1 & \text{if } r = q \\ \mu_2(q) & \text{if } r \in X \text{ and } q \in Q_2 \\ 0 & \text{otherwise.} \end{cases}$$

We first need to show that A is a well-defined substitution summable PTA. By definition of μ and δ one sees that μ and $\delta(q, f)$, for $q \in Q_2$, are distributions. Let $q \in Q_1 \setminus X$ and $f \in \Sigma$:

$$\begin{aligned}
&\sum_{q_1,\dots,q_n \in Q} \delta(q, f)(q_1, \dots, q_n) \\
&= \sum_{q_1,\dots,q_n \in Q} \sum_{r_1,\dots,r_n \in Q_1} \delta_1(q, f)(r_1, \dots, r_n) \prod_{i=1}^{n} \kappa(r_i, q_i) \\
&= \sum_{r_1,\dots,r_n \in Q_1} \delta_1(q, f)(r_1, \dots, r_n) \prod_{i=1}^{n} \sum_{q \in Q} \kappa(r_i, q)
\end{aligned}$$

By definition, κ is either $\mathbb{1}_{\{r_i\}}$ or μ_2, depending on r_i:

$$\begin{aligned}
&= \sum_{r_1,\dots,r_n \in Q_1} \delta_1(q, f)(r_1, \dots, r_n) \prod_{i=1}^{n} \begin{cases} 1 & \text{if } r_i \in Q_1 \setminus X \\ \mu_2(Q_2) & \text{if } r_i \in X \end{cases} \\
&= \sum_{r_1,\dots,r_n \in Q_1} \delta_1(q, f)(r_1, \dots, r_n) \\
&\leq 1.
\end{aligned}$$

Next, we show that A actually has the behaviour $\|A_1\| \cdot_z \|A_2\|$. To employ induction on the height of the input tree, we prove a slightly stronger statement:

$$\delta_q(t) = \big((\delta_1)_q \cdot_z \|A_2\|\big)(t) \tag{7.3}$$

for all $t \in \mathrm{T}_\Sigma(V)$ with and $q \in Q_1 \setminus X$, where δ_q is defined as below Definition 2.25. Let $q \in Q_1 \setminus X$, $t \in \mathrm{T}_\Sigma(V)$ and assume $\mathrm{height}(t) = 0$, i.e, $t = a \in \Sigma_0 \cup V$. We have $\delta_q(t) = \mathbb{1}_{F_{\{a\}}}(q)$. In case $a = z$, $(q, z) \notin F$ by construction of F. Thus, $\delta_q(t) = 0 = ((\delta_1)_q \cdot_z \|A_2\|)(t)$ since $(\delta_1)_q(z) = 0$ by choice of $q \in Q_1 \setminus X$. In case $a \neq z$, we have $(q, a) \in F$ iff $(q, a) \in F_1$. Thus, $\delta_q(t) = (\delta_1)_q(t) = (\delta_1)_q(t) + (\delta_1)_q(z)\, \|A_2\|(t)$, again, as $(\delta_1)_q(z) = 0$.

Now, consider a tree $t = f(t_1, \dots, t_n)$ with $n \geq 1$. We obtain

$$\begin{aligned}
\delta_q(t) &= \sum_{q_1,\dots,q_n \in Q} \delta(q, f)(q_1, \dots, q_n) \prod_{i=1}^{n} \delta_{q_i}(t_i) \\
&= \sum_{q_1,\dots,q_n \in Q} \sum_{r_1,\dots,r_n \in Q_1} \delta_1(q, f)(r_1, \dots, r_n) \prod_{i=1}^{n} \kappa(r_i, q_i)\, \delta_{q_i}(t_i) \\
&= \sum_{r_1,\dots,r_n \in Q_1} \delta_1(q, f)(r_1, \dots, r_n) \prod_{i=1}^{n} \sum_{q \in Q} \kappa(r_i, q)\, \delta_q(t_i)
\end{aligned}$$

$$= \sum_{r_1,\dots,r_n \in Q_1} \delta_1(q,f)(r_1,\dots,r_n) \prod_{i=1}^{n} \begin{cases} \delta_{r_i}(t_i) & \text{if } r_i \in Q_1 \setminus X \\ \sum_{q \in Q_2} \mu_2(q)\, \delta_q(t_i) & \text{if } r_i \in X \end{cases}$$

By induction hypothesis, we have $\delta_{r_i} = (\delta_1)_{r_i} \cdot_z \|A_2\|$. Moreover, $\sum_{q \in Q_2} \mu_2(q)\, \delta_q(t_i) = \|A_2\|(t_i)$. Finally, remark that $(\delta_1)_{r_i} = \mathbb{1}_{\{z\}}$ if $r_i \in X$, as every state in X is a sink state. Altogether we obtain

$$= \sum_{r_1,\dots,r_n \in Q_1} \delta_1(q,f)(r_1,\dots,r_n) \prod_{i=1}^{n} \big((\delta_1)_{r_i} \cdot_z \|A_2\|\big)(t_i)$$

By the upcoming Proposition 7.23 we obtain the following since $(\delta_1)_q(z) = 0$:

$$= \big((\delta_1)_q \cdot_z \|A_2\|\big)(t)$$

This shows (7.3). Finally, we deduce the desired statement about $\|A\|$:

$$\|A\|(t) = \sum_{q \in Q} \mu(q)\, \delta_q(t) = \sum_{q \in Q_1 \setminus X} \mu_1(q) \big((\delta_1)_q \cdot_z \|A_2\|\big)(t) + \sum_{q \in Q_2} \mu_1(X)\, \mu_2(q)\, (\delta_2)_q(t)$$

$$= \sum_{q \in Q_1 \setminus X} \left(\mu_1(q) \sum_{s \trianglelefteq_z t} (\delta_1)_q(s) \prod_{x \in \mathrm{pos}_z(s)} \|A_2\|(t|_x) \right) + \|A_1\|(z)\, \|A_2\|(t).$$

Since $q \notin X$ in the first sum, $(\delta_1)_q(z) = 0$ as X contains all final states for z. Thus, we only need to consider trees $s \neq z$.

$$= \sum_{z \neq s \trianglelefteq_z t} \|A_1\|(s) \prod_{x \in \mathrm{pos}_z(s)} \|A_2\|(t|_x) + \|A_1\|(z)\, \|A_2\|(t)$$

$$= (\|A_1\| \cdot_z \|A_2\|)(t).$$

This shows the correctness of the construction of A and therefore completes the proof. ∎

The next statement is an auxiliary result, which allows us to decompose products of the form $\delta_q \cdot_z S$.

Proposition 7.23. Let $z \in V$, $A = (Q, \delta, \mu, F)$ a substitution summable probabilistic tree automaton, $q \in Q$ a state with $(q,z) \notin F$, and S a tree series. We have

$$\big(\delta_q \cdot_z S\big)(t) = \sum_{q_1,\dots,q_n \in Q} \delta(q,f)(q_1,\dots,q_n) \prod_{i=1}^{n} \big(\delta_{q_i} \cdot_z S\big)(t_i),$$

for all $t = f(t_1,\dots,t_n)$ with $n \geq 1$.

Proof. Let z, A, S, and t as in the statement of the lemma. We show the statement by direct computation. Since $(q, z) \notin F$, we have $\delta_q(z) = 0$:

$$(\delta_q \cdot_z S)(t) = \sum_{z \neq s \trianglelefteq_z t} \delta_q(s) \prod_{x \in \mathrm{pos}_z(s)} S(t|_x)$$

Every tree s with $z \neq s \trianglelefteq_z t$ is of the form $s = f(s_1, \ldots, s_n)$ with $s_i \trianglelefteq_z t_i$ for all $i = 1, \ldots, n$. Hence

$$= \sum_{\substack{s_i \trianglelefteq_z t_i \\ (i=1,\ldots,n)}} \sum_{q_1,\ldots,q_n \in Q} \delta(q, f)(q_1, \ldots, q_n) \prod_{i=1}^{n} \delta_{q_i}(s_i) \prod_{x \in \mathrm{pos}_z(s_i)} S(t_i|_x)$$

$$= \sum_{q_1,\ldots,q_n \in Q} \delta(q, f)(q_1, \ldots, q_n) \prod_{i=1}^{n} (\delta_{q_i} \cdot_z S)(t_i). \qquad \blacksquare$$

Lemma 7.24. Let $z \in V$ and A be a substitution summable probabilistic tree automaton. There is a substitution summable probabilistic tree automaton A' with $\|A'\| = \|A\|^{\infty z}$.

Proof. The proof is similar to the proof of Lemma 7.22. Instead of redirecting transition from a state in F_z to the initial states of A_2, the transitions are redirected to enter A again. Additionally, the probabilities of these transitions are multiplied by a factor λ to model arbitrarily many substitutions of z by itself.

Assume $A = (Q, \delta, \mu, F)$ and let $X = \{q \in Q \mid (q, z) \in F\}$. We may assume $\mu(X) < 1$. Otherwise, $\|A\| = \mathbb{1}_{\{z\}}$ as A is substitution summable and $\mathbb{1}_{\{z\}}{}^{\infty} = \mathbb{1}_{\{z\}}$. Let $A' = (Q', \delta', \mu', F')$ be given by

$$Q' = Q \setminus X, \qquad F' = F \setminus (X \times \{z\}), \qquad \mu'(q) = \lambda\, \mu(q)$$

$$\delta'(q, f)(q_1, \ldots, q_n) = \sum_{r_1,\ldots,r_n \in Q} \delta(q, f)(r_1, \ldots, r_n) \prod_{i=1}^{n} \kappa(r_i, q_i),$$

where

$$\lambda = \frac{1}{1 - \mu(X)} \qquad \text{and} \qquad \kappa(r, q) = \begin{cases} \mathbb{1}_{\{r\}}(q) & \text{if } r \in Q' \\ \lambda\, \mu(q) & \text{if } r \in X. \end{cases}$$

Before we show the correctness of the construction, we prove the following auxiliary equation, which will allow us to use induction on the tree height, for all $t = f(t_1, \ldots, t_n)$ with $n > 0$:

$$\|A\|^{\infty z}(t) = \sum_{q \in Q'} \lambda\, \mu(q)\, (\delta_q \cdot_z \|A\|^{\infty z})(t). \tag{7.4}$$

As X is disjoint from any other set $F_{\{a\}} = \{q \in Q \mid (q,a) \in F\}$ for $a \neq z$ and every state in X is a sink state in A by substitution summability of A, we have $\sum_{q\in Q'} \mu(q)\,\delta_q(t) = \|A\|(t)$ for any tree $t \neq z$, and $\delta_q(z) = 0$ for all $q \in Q'$. Therefore

$$\begin{aligned}\sum_{q\in Q'} \lambda\,\mu(q)\big(\delta_q \cdot_z \|A\|^{\circ z}\big)(t) &= \lambda \sum_{z\neq s \trianglelefteq_z t} \|A\|(s) \prod_{x\in \mathrm{pos}_z(s)} \|A\|^{\circ z}(t|_x) \\ &= \lambda\,\big((\|A\| - \|A\|(z)\,\mathbb{1}_{\{z\}}) \cdot_z \|A\|^{\circ z}\big)(t).\end{aligned}$$

Let S be an arbitrary substitution summable tree series. As $S \cdot_z S^{\circ z} = S^{\circ z}$ by Lemma 7.12, we obtain

$$(S - S(z)\,\mathbb{1}_{\{z\}}) \cdot_z S^{\circ z} = S \cdot_z S^{\circ z} - S(z)\,S^{\circ z} = (1 - S(z))\,S^{\circ z}.$$

Therefore, we obtain the following for $S = \|A\|$:

$$\|A\|^{\circ z} = \frac{(\|A\| - \|A\|(z)\,\mathbb{1}_{\{z\}}) \cdot_z \|A\|^{\circ z}}{1 - \|A\|(z)}.$$

As $\lambda = \frac{1}{1-\mu(X)} = \frac{1}{1-\|A\|(z)}$, this shows (7.4).

We are now ready to prove $\|A'\| = \|A\|^{\circ}$. We show

$$\delta'_q(t) = \big(\delta_q \cdot_z \|A\|^{\circ z}\big)(t) \tag{7.5}$$

for all $q \in Q'$ and $t \in \mathrm{T}_\Sigma(V)$ using induction on height(t). Let $t = a \in \Sigma_0 \cup V$. We obtain $\delta'_q(a) = 0$ if $a = z$ and, in case $a \neq z$, $\delta'_q(a) = \mathbb{1}_{F'}((q,a)) = \mathbb{1}_F((q,a)) = \delta_q(t) = \delta_q(t) + \delta_q(z)\,\|A\|^{\circ z}(t)$, as $\delta_q(z) = 0$ by the choice of q.

Next, assume $t = f(t_1,\dots,t_n)$ with $n \geq 1$. We compute

$$\|\delta'_q\|(t) = \sum_{q_1,\dots,q_n\in Q'} \delta'(q,f)(q_1,\dots,q_n) \prod_{i=1}^{n} \delta'_{q_i}(t_i)$$

By induction hypothesis, $\delta'_{q_i}(t_i) = (\delta_{q_i} \cdot_z \|A\|^{\circ})(t_i)$ for $i = 1,\dots,n$:

$$= \sum_{q_1,\dots,q_n\in Q'} \sum_{r_1,\dots,r_n\in Q} \delta(q,f)(r_1,\dots,r_n) \prod_{i=1}^{n} \kappa(r_i,q_i)\,(\delta_{q_i} \cdot_z \|A\|^{\circ z})(t_i)$$

We apply distributivity, and insert the definition of κ:

$$= \sum_{r_1,\dots,r_n\in Q} \delta(q,f)(r_1,\dots,r_n) \cdot \prod_{i=1}^{n} \Bigg(\mathbb{1}_X(r_i) \underbrace{\sum_{q\in Q'} \lambda\,\mu(q)\,(\delta_q \cdot_z \|A\|^{\circ z})(t_i)}_{\overset{(7.4)}{=} \|A\|^{\circ z}(t_i)} + \mathbb{1}_{Q'}(r_i)\,(\delta_{r_i} \cdot_z \|A\|^{\circ z})(t_i) \Bigg)$$

Since $\delta_{r_i} = \mathbb{1}_{\{z\}}$ for $r_i \in X$, we obtain $\|A\|^{\infty z} = \delta_{r_i} \cdot_z \|A\|^{\infty z}$. We conclude

$$= \sum_{r_1,\ldots,r_n \in Q} \delta(q, f)(r_1, \ldots, r_n) \prod_{i=1}^{n} \big(\delta_{r_i} \cdot_z \|A\|^{\infty z}\big)(t_i)$$

Again, we apply Proposition 7.23 using that $\delta_q(z) = 0$:

$$= \big(\delta_q \cdot_z \|A\|^{\infty z}\big)(t).$$

Therefore, we obtain (7.5). We use (7.4) to show that A' indeed has the desired behaviour:

$$\|A'\| = \sum_{q \in Q'} \lambda\, \mu(q)\, \delta'_q(t) = \sum_{q \in Q'} \lambda\, \mu(q) \big(\delta_q \cdot_z \|A\|^{\infty z}\big) \overset{(7.4)}{=} \|A\|^{\infty z}.$$

This completes the proof. ∎

Combining the results from this section, we have proven the following lemma.

Lemma 7.25. Let E be a probabilistic regular tree expression. There is a probabilistic tree automaton A with $\|E\| = \|A\|$.

Proof. Let $M = \{E \in \mathrm{PRTE} \mid \exists \text{substitution summable PTA } A\colon \|A\| = \|E\|\}$. Clearly, $\mathbb{0} \in M$ and $\mathbb{1}_z \in M$ for all $z \in V$. By Lemmas 7.20 to 7.22 and 7.24 and the fact that application of the associativity, commutativity, and distributivity does not change the semantics of an expression, M satisfies the closure properties of Definition 7.15. Thus, $\mathrm{PRTE} = M$. ∎

7.5 From Automata to Expressions

This subsection contains the proof of the following lemma. As the proof is one monolithic argument, it uses all of this section.

Lemma 7.26. Let A be a top-down probabilistic tree automaton over T_Σ. There is a set of variables V and a probabilistic regular tree expressions E over $\mathrm{T}_\Sigma(V)$ such that $\|A\| = \|E\|$.

Let $A = (Q, \delta, \mu, F)$. We set $V = Q$. The idea of the proof is similar to the classical case, but additional care has to be taken to handle the syntax restrictions of PRTE. Let $X \subseteq Q$ and $t \in \mathrm{T}_\Sigma(X)$ with $t(\varepsilon) \notin X$. We define the following sets of runs over t: let $R_q^X(t)$ contain runs $\rho\colon \mathrm{pos}(t) \to Q$ with

1. $\rho(\varepsilon) = q$,
2. $\rho(x) \in Q \setminus X$ for all $x \in \mathrm{pos}_\Sigma(t) \setminus \{\varepsilon\}$,
3. $\rho(x) = t(x)$ for all $x \in \mathrm{pos}_X(t)$,
4. $(\rho(x), t(x)) \in F$ for all $x \in \mathrm{pos}_{\Sigma_0}(t)$.

Intuitively, the runs in $R_q^X(t)$ must start at q and may only attain states from X if t is also labelled with a state from X at this position, and the two states must match. At all positions, where t is labelled with some letter from Σ, only states from $Q \setminus X$are allowed in the runs. Furthermore, all leaf nodes not labelled by states in t must satisfy the acceptance condition of A.

With the help of $R_q^X(t)$, we define the tree series S_q^X on $\mathrm{T}_\Sigma(Q)$ by

$$S_q^X(t) = \begin{cases} \displaystyle\sum_{\rho \in R_q^X(t)} \prod_{x \in \mathrm{inner}(t)} \delta(\rho(x), t(x))\big(\rho(x1), \ldots, \rho(x\,arity(t(x)))\big) & \\ & \text{if } t \in \mathrm{T}_\Sigma(X) \setminus X \\ 0 & \text{otherwise.} \end{cases}$$

Since, for a tree $t \in \mathrm{T}_\Sigma$, $R_q^Q(t)$ is the set of all runs of A on t starting in q, we have $\|A\|(t) = \sum_{q \in Q} \mu(q)\, S_q^\emptyset(t)$. We will construct expressions E_q^X such that $\|E_q^X\|(t) = S_q^X(t)$ if $t \in \mathrm{T}_\Sigma(X) \setminus X$ and $\|E_q^X\|(t) = 0$ otherwise using induction on $|Q \setminus X|$.

First, we consider the case $X = Q$. By condition 2, we have $R_q^X(t) \neq \emptyset$ only for trees of the form $t = f(q_1, \ldots, q_n)$ with $f \in \Sigma_n$ and $q_i \in Q$. In this case $R_q^X(t)$ contains the single run with q at the root node and $q_1, \ldots, q_n$ at the child nodes. Thus, $S_q^X(t) = \delta(q, f)(q_1, \ldots, q_n)$. We construct the expression E_q^Q as follows:

$$E_q^Q = \sum_{\substack{a \in \Sigma_0 \\ (q,a) \in F}} a + \sum_{\substack{f \in \Sigma_n \\ n \geq 1}} \sum_{q_1, \ldots, q_n \in Q} \delta(q, f)(q_1, \ldots, q_n) \cdot f(q_1, \ldots, q_n).$$

By Definition 7.15 (2) every state q_i is an expression. Hence, E_q^Q is also an expression by the following Sublemma 7.27.

Sublemma 7.27. For every $f \in \Sigma$ let X_f be a finite set and $(\lambda_x^f)_{x \in X_f}$ be a distribution on X_f. Furthermore, let $e_f \colon X_f \to \mathrm{PRTE}^{arity(f)}$ be a mapping for every $f \in \Sigma$. Then,

$$E = \sum_{f \in \Sigma} \sum_{x \in X_f} \lambda_x^f f(e_f(x))$$

is also a probabilistic regular tree expression.

Proof. Let $\mathfrak{X}$ contain all function $s\colon \Sigma \to \bigcup_{f\in\Sigma} X_f$ with $s(f) \in X_f$ for all $f \in \Sigma$. Then, the function $s \mapsto \prod_{g\in\Sigma} \lambda^g_{s(g)}$ is a distribution on $\mathfrak{X}$. We define the expression E' by

$$E' = \sum_{s\in\mathfrak{X}} \left(\prod_{g\in\Sigma} \lambda^g_{s(g)}\right) \sum_{f\in\Sigma} f\big(e_f(s(f))\big).$$

We have that $\sum_{f\in\Sigma} f(e_f(s(f)))$ is a PRTE for every $s \in \mathfrak{X}$ by Definition 7.15 (3). Hence, E' is also an PRTE by iterated application of Definition 7.15 (4). We still need to show that $E = E'$, i.e., that E' can be transformed into E by application of the ACD rules:

$$\begin{aligned}
E' &\equiv \sum_{f\in\Sigma} \sum_{s\in\mathfrak{X}} \left(\prod_{g\in\Sigma} \lambda^g_{s(g)}\right) f\big(e_f(s(f))\big) \\
&\equiv \sum_{f\in\Sigma} \sum_{\substack{s_g\in X_g \\ (g\in\Sigma)}} \left(\prod_{g\in\Sigma} \lambda^g_{s(g)}\right) f\big(e_f(s_f)\big) \\
&\equiv \sum_{f\in\Sigma} \sum_{s_f\in X_f} \left(\sum_{\substack{s_g\in X_g \\ (g\in\Sigma,\, g\neq f)}} \prod_{g\in\Sigma} \lambda^g_{s_g}\right) f\big(e_f(s_f)\big) \\
&\equiv \sum_{f\in\Sigma} \sum_{s_f\in X_f} \left(\lambda^f_{s_f} \prod_{g\in\Sigma\setminus\{f\}} \sum_{s_g\in X_g} \lambda^g_{s_g}\right) f\big(e_f(s_f)\big) \\
&\equiv \sum_{f\in\Sigma} \sum_{s_f\in X_f} \lambda^f_{s_f} f\big(e_f(s_f)\big) \\
&\equiv E.
\end{aligned}$$

This completes the proof of Sublemma 7.27. ∎

Next, we consider the case $|Q \setminus X| > 0$, i.e., $Q \setminus X \neq \emptyset$. Let $q \in Q \setminus X$ be some fixed state. Let $X' = X \cup \{q\}$. We have $|Q \setminus X'| < |Q \setminus X|$. Thus, there are expressions $E_p^{X'}$ for all $p \in Q$ with $\|E_p^{X'}\|(t) = S_p^{X'}(t)$ for all $t \in \mathrm{T}_\Sigma(Q)$. We show $S_p^X = S_p^{X'} \cdot_q S_q^X$ for all $p, q \in Q$. In order to prove this statement, we show how a set of runs can be decomposed in a top parts only containing q at the leaf nodes, and a bottom part, where q may occur anywhere. For any run ρ on t let $\min_q(\rho)$ be the set of prefix-minimal positions in $\mathrm{pos}(t)$ with $\rho(x) = q$. Recall that a $\preceq$-antichain is a set $M \subseteq \mathrm{pos}(t)$ such that $x \preceq y$ implies $x = y$ for all $x, y \in M$.

Sublemma 7.28. Let $M \subseteq \mathrm{pos}(t)$ be a $\preceq$-antichain. We define a function g on the set of all runs with $\rho \in R_p^X(t)$ with $\min_q(\rho) = M$ by $g(\rho) = (\rho|_{\mathrm{pos}(t)\setminus M\mathbb{N}^+}, (\rho|_x)_{x\in M})$, where $\rho|_{\mathrm{pos}(t)\setminus M\mathbb{N}^+} \colon \mathrm{pos}(t) \setminus M\mathbb{N}^+ \to Q$ is the restriction of ρ to all positions above of M or incomparable to M, and $\rho|_x \colon \{y \in \mathbb{N}^* \mid xy \in \mathrm{pos}(t)\} \to Q$ the run below the position x.

Then, g is a bijection from $\{\rho \in R_p^X(t) \mid \min_q(\rho) = M\}$ to the set of tuples $(\rho, (\rho_x)_{x\in M})$ with $\rho \in R_p^{X'}(t[M \leftarrow q])$ and $\rho_x \in R_q^X(t|_x)$ for all $x \in M$.

Proof. We verify that g is well-defined. Given a tree $t \in \mathrm{T}_\Sigma(X)$, a position $x \in M$, and a run $\rho \in R_t^X(t)$, we clearly have $\rho|_x \in R_q^X(t|_x)$ as $t(x) = q$. Moreover, as the positions in M are minimal the state q occurs only at leaf nodes in the run $\rho' = \rho|_{\mathrm{pos}(t)\setminus M\mathbb{N}^+}$. These are exactly the positions where $t[M \leftarrow q]$ is labelled by q. Thus, $\rho' \in R_q^{X'}(t[M \leftarrow q])$.

We show that the function $h(\rho, (\rho_x)_{x\in M}) = \rho[x \leftarrow \rho_x]_{x\in M}$, where $\rho \in R_p^{X'}(t[M \leftarrow q])$ and $\rho_x \in R_q^X(t|_x)$ for all $x \in M$, is the inverse of g. The well-definedness of h follows directly from the definition of the sets $R_p^{X'}(t[M \leftarrow q])$ and $R_q^X(t|_x)$.

Let $(g \circ h)(\rho, (\rho_x)_{x\in M}) = (\rho', (\rho'_x)_{x\in M})$ and $h(\rho, (\rho_x)_{x\in M}) = \tau$. As the underlying trees of the runs coincide, we have $\mathrm{pos}(\rho) = \mathrm{pos}(\rho')$ and $\mathrm{pos}(\rho_x) = \mathrm{pos}(\rho'_x)$ for all $x \in M$. Let $y \in \mathrm{pos}(\rho)$. If $y \notin M$, we have $\rho(y) = \rho[x \leftarrow \rho_x]_{x\in M}(y) = \tau(y) = \tau|_{\mathrm{pos}(t)\setminus M\mathbb{N}^+}(y) = \rho'(y)$, and for $y \in M$ we obtain $\rho(y) = t[M \leftarrow q](y) = q$ and $\rho'(y) = t[M \leftarrow q](y) = q$. Hence, $\rho = \rho'$. Next, let $x \in M$ and $y \in \mathrm{pos}(\rho_x)$. We conclude $\rho_x(y) = \rho[x \leftarrow \rho_x]_{x\in M}(xy) = \tau(xy) = \tau|_x(y) = \rho'_x(y)$. Thus, $\rho_x = \rho'_x$. This implies that $g \circ h = \mathrm{id}$.

Conversely, assume $(h \circ g)(\tau) = \tau'$ and let $g(\tau) = (\rho, (\rho_x)_{x\in M})$. Let $y \in \mathrm{pos}(t)$ be not below any position in M. We conclude $\tau(y) = \tau_{\mathrm{pos}(t)\setminus M\mathbb{N}^+}(y) = \rho(y) = \rho[x \leftarrow \rho_x]_{x\in M}(y) = \tau'(y)$. Now, assume $y = xy'$ for some $x \in M$. Hence, $\tau(y) = \tau|_x(y') = \rho_x(y') = \rho[x \leftarrow \rho_x]_{x\in M}(xy') = \tau'(y)$. Thus, $\tau = \tau'$ and $h \circ g = \mathrm{id}$. This shows that $h = g^{-1}$ and g is bijective. ∎

We use the statement of Sublemma 7.28 to show $S_p^X = S_p^{X'} \cdot_q S_q^X$ for all $p, q \in Q$. Let $t \in \mathrm{T}_\Sigma(Q)$ arbitrary. If $t \notin \mathrm{T}_\Sigma(X) \setminus X$ then either $t = r \in X$ or $t(x) \in Q \setminus X$ for some $x \in \mathrm{pos}(t)$. In the first case, we have $s \in Q$ for all $s \trianglelefteq_q t$. Thus, $S_p^{X'}(s) = 0$ and so $(S_p^{X'} \cdot_q S_q^X)(t) = 0$. In the second case, we have either $t(x) \in Q \setminus X'$ or $t(x) = q$ for some $x \in \mathrm{pos}(t)$. If $t(x) \in Q \setminus X'$, then for every $s \trianglelefteq_z t$ either $\mathrm{pos}_{Q\setminus X'}(s) \neq \emptyset$ or $\mathrm{pos}_{Q\setminus X'}(t|_x) \neq \emptyset$ for some $x \in \mathrm{pos}_q(s)$. Hence, $S_p^{X'}(s) \prod_{x\in \mathrm{pos}_q(s)} S_q^{X'}(t|_x) = 0$ and so $(S_p^{X'} \cdot_z S_q^X)(t) = 0$. In the latter case, i.e., $t(x) = q$ for some $x \in \mathrm{pos}(t)$, consider a tree $s \trianglelefteq_z t$. By definition of $\trianglelefteq_z$, there is an $x' \in \mathrm{pos}_q(s)$ with $x' \preceq x$. Since then $\mathrm{pos}_q(t|_{x'}) \neq \emptyset$ and $S_q^X(T|_x) = 0$, we conclude $(S_p^{X'} \cdot_q S_q^X)(t) = 0$.

Now, assume $t \in \mathrm{T}_\Sigma(X) \setminus X$. We compute

$$S_p^X(t) = \sum_{\rho \in R_p^X(t)} \prod_{x \in \mathrm{inner}(t)} \delta(\rho(x), t(x))\big(\rho(x1), \ldots, \rho(x\mathit{arity}(t(x)))\big)$$

Each run $\rho \in R_p^X(t)$ as a unique set $\min_q(\rho)$ of minimal positions labelled with q:

$$= \sum_{M \subseteq \mathrm{pos}(t) \text{ antichain}} \sum_{\substack{\rho \in R_p^X(t) \\ \min_q(\rho) = M}} \prod_{x \in \mathrm{inner}(t)} \delta(\rho(x), t(x))(\rho(x1), \ldots, \rho(xn_x)),$$

where $n_x = \mathit{arity}(t(x))$. We apply Sublemma 7.28 to the index set of the summation:

$$= \sum_{M \subseteq \mathrm{pos}(t) \text{ antichain}} \sum_{\substack{\rho \in R_p^{X'}(t[M \leftarrow q]) \\ \rho_x \in R_q^X(t|_x)\ (x \in M)}} \cdot \prod_{x \in \mathrm{inner}(\rho[x \leftarrow \rho_x]_{x \in M})} \delta(\rho[x \leftarrow \rho_x]_{x \in M}(x), t(x))(\rho[x \leftarrow \rho_x]_{x \in M}(x1), \ldots)$$

Next, we split the product in the inner positions below any element of M and the inner positions above or incomparable to M. Note that the latter set equals $\mathrm{inner}(t[M \leftarrow q])$:

$$= \sum_{M \subseteq \mathrm{pos}(t) \text{ antichain}} \sum_{\substack{\rho \in R_p^{X'}(t[M \leftarrow q]) \\ \rho_x \in R_q^X(t|_x)\ (x \in M)}} \cdot \prod_{x \in \mathrm{inner}(t[M \leftarrow q])} \delta(\rho(x), t(x))(\rho(x1), \ldots, \rho(xn_x)) \cdot \prod_{y \in M} \prod_{x \in \mathrm{inner}(t|_y)} \delta(\rho_y(x), t|_y(x))(\rho_y(x1), \ldots, \rho_y(xn_x))$$

Using distributivity, we obtain the definitions of $S_p^{X'}$ and S_q^X, respectively:

$$= \sum_{M \subseteq \mathrm{pos}(t) \text{ antichain}} S_p^{X'}(t[M \leftarrow q]) \prod_{y \in M} S_q^X(t|_y).$$

Finally, note that the mapping $M \mapsto t[M \leftarrow q]$ is a bijection between the antichains in $\mathrm{pos}(t)$ and the trees $s \in \mathrm{T}_\Sigma(X')$ with $s \trianglelefteq_q t$ as q does not appear as label in t. The inverse function of this bijection is $s \mapsto \mathrm{pos}_q(s)$. Thus, we continue:

$$= \sum_{s \trianglelefteq_q t} S_p^{X'}(s) \prod_{y \in \mathrm{pos}_q(s)} S_q^X(t|_y)$$

$$= \left(S_p^{X'} \cdot_z S_q^X\right)(t).$$

From $S_p^X = S_p^{X'} \cdot_q S_q^X$ we directly conclude two statements: First, when setting $p = q$, we obtain $S_q^X = S_q^{X'} \cdot_q S_q^X$ and therefore $S_q^X = (S_q^{X'})^{\infty q}$ by Lemma 7.12, and so we conclude $S_p^X = S_p^{X'} \cdot_q (S_q^{X'})^{\infty q}$. Thus, by induction hypothesis, $S_p^X = \|E_p^{X'}\| \cdot_q \|E_q^{X'}\|^{\infty z}$. We define $E_p^X = E_p^{X'} \cdot_q (E_q^{X'})^{\infty q}$ for every $p \in Q$. This definition satisfies $\|E_p^X\| = S_p^X$, by the above calculation. Hence, we have completed the inductive construction of the expressions E_p^X.

We are now ready to define the expression E with $\|E\| = \|A\|$ using the case $X = \emptyset$:

$$E = \sum_{q \in Q} \mu(q)\, E_q^{\emptyset}.$$

This is a valid expression by iterated application of Definition 7.15 (4). As the sets $R_q^{\emptyset}(t)$ contain exactly the successful runs of A on t starting in q, we have $\|E\| = \|A\|$ as claimed. This completes the proof of Lemma 7.26. ∎

With results of this section and the last section, we finally have proven the following theorem:

Theorem 7.29. Let Σ be a rank alphabet and $S\colon \mathrm{T}_\Sigma \to [0,1]$ a probabilistic tree series. The following statements are equivalent:

1. $S = \|A\|$ for a top-down probabilistic tree automaton A.
2. $S = \|E\|$ for a probabilistic regular tree expression over some set of variables.

The constructions in both directions are effective.

Part III

Model Checking LTL over the Infinite Tree

Chapter 8

Constraint LTL and Constraint Büchi Automata

In this chapter, we introduce Constraint Linear Temporal Logic, or cLTL for short, a variant of Linear Temporal Logic (LTL) with local constraints. A model of a formula of this logic is a (multi-) data word with data values from some $\{\preceq, \sqsubseteq, S\}$-structure. We are particularly interested in the case where this structure is an ordered tree with prefix order $\preceq$ and lexicographic order $\sqsubseteq$. Our goal of this chapter and the next chapter is to adjust the automata-based model checking methods known for LTL to this setting.

For this purpose, we first introduce our notion of the infinite tree in Section 8.1 and Constraint LTL in Section 8.2. Afterwards we recall constraint automata in Section 8.3. As last part of this chapter we prove in Section 8.4 that satisfiability and model checking for cLTL formulas with constraints over the full infinitely branching tree are in PSPACE due to a reduction to the emptiness problem of tree-constraint automata. The technical core for containment in PSPACE is to show that emptiness of tree-constraint automata is PSPACE-complete and NL-complete for fixed dimension. The proof of this result is postponed to Chapter 9. We conclude this chapter by providing a reduction of the satisfiability and model checking problem for cLTL over the full finitely branching tree or over the trees with branching structure ω or $-\omega + \omega$ to the corresponding problem over the full infinitely branching tree.

This is joint work with Alexander Kartzow. The results can also be found in [KW15].

8.1 Data Words over the Infinite Tree

Let us first give an exact definition of "infinite tree". For this, we extend the notion of signature introduced in Definition 4.1 to signatures with constants. Special treatment of constants was not necessary in Chapter 4 as a constant can be simulated

in MSO logic using existential quantification and a unary predicate which is interpreted as a singleton set. Formally, a *signature with constants* or just a signature is a pair $\mathcal{S} = (S, arity)$ such that S is a set and $arity\colon S \to \mathbb{N}_0$ is a function. A $\mathcal{S}$-structure is a tuple $\mathcal{A} = (A, (R^{\mathcal{A}})_{R\in S})$ such that A is any set and $R^{\mathcal{A}} \subseteq A^{arity(R)}$ if $arity(R) > 0$ and $R^{\mathcal{A}} \in A$ if $arity(R) = 0$ for every $R \in S$.

Next, we choose a set $D \subseteq \mathbb{Q}$ which will describe the branching structure of the tree, i.e., the order structure of the child nodes of any node. We consider the cases $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}\}$ which describe infinitely branching trees, and $D = \{1, \dots, k\}$, for some $k \geq 1$, for finitely branching trees. Note that though the branching degree may be finite, we always consider trees of infinite height, i.e., trees without leaf nodes. Furthermore, we introduce a finite number of constants symbols $s_1, \dots, s_m$ to mark distinguished nodes in the tree. Intuitively, the tree $\mathcal{T}_D^C$ is the unlabelled infinite tree where the children of every node are ordered like $(D, \leq)$ and nodes can be compared using prefix order and lexicographic order, i.e., left-right order. Additionally, a finite set of nodes is distinguishably marked as constants.

Formally, let D be one of the above sets and $C = \{c_1, \dots, c_m\} \subseteq D^*$ a set of constants. Let the signature σ be given by $\sigma = \{\preceq, \sqsubseteq, s_1, \dots, s_m\}$ where $s_1, \dots, s_m$ are constants symbols. The *infinite tree over D with constants C* is the σ-structure $\mathcal{T}_D^C$ given by

$$\mathcal{T}_D^C = (D^*, \preceq_D, \sqsubseteq_D, c_1, c_2, \dots, c_m),$$

where $\preceq_D$ is the prefix order on D^*, and $\sqsubseteq_D$ is the lexicographic order on D^* with respect to the natural order on D. Note that, apart from the constants, there is no labelling on the tree $\mathcal{T}_D^C$. If D is understood, we just write $\preceq$ and $\sqsubseteq$. For $D = \{1, \dots, k\}$ we also write $\mathcal{T}_k^C$ instead of $\mathcal{T}_D^C$.

In order to reason about elements of $\mathcal{T}_D^C$, i.e., positions in the tree, using automata or temporal logic, we use data words over D^*. Intuitively, a data word is an infinite word, where a fixed number of elements of D^* replaces the symbols of a finite alphabet. Formally, given a σ-structure $\mathcal{A} = (A, \preceq^{\mathcal{A}}, \sqsubseteq^{\mathcal{A}} s_1^{\mathcal{A}}, s_2^{\mathcal{A}}, \dots, s_m^{\mathcal{A}})$, an *$n$-dimensional data word over $\mathcal{A}$* is any element of $\left(A^n\right)^\omega$.

For two positions $x, y \in D^*$ let $x \sqcap y$ be the *maximal common prefix of x and y*, i.e., $z = x \sqcap y$ if $z \in D^*$ is the $\preceq$-maximal position with $z \preceq x$ and $z \preceq y$. This position always exists, as ε is always a possible choice and the finitely many prefixes of x (or equivalently y) are linearly ordered by $\preceq$.

8.2 LTL with Constraints

Constraint LTL has been introduced by Demri and D'Souza [DD07] for arbitrary domains. Here, as we are only interested in the case of trees with prefix order and

lexicographic order, we recall the definition of Constraint LTL for this signature only.

The logic *Constraint LTL* over the signature $\sigma = \{\preceq, \sqsubseteq, s_1, s_2, \ldots, s_m\}$ where $S = \{s_1, \ldots, s_m\}$ is a set of constant symbols, abbreviated cLTL, is given by the grammar

$$\varphi ::= \mathrm{X}^i\, x_1 \sim s \mid s \sim \mathrm{X}^i\, x_1 \mid \mathrm{X}^i\, x_1 \sim \mathrm{X}^j\, x_2 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathrm{X}\,\varphi \mid \varphi\,\mathrm{U}\,\varphi,$$

where $\sim \in \{=, \preceq, \sqsubseteq\}$, i, j are non-negative integers, x_1, x_2 are variables from some countable fixed set $\mathcal{V}$ and $s \in S$ is a constant symbol. Note that "X^i" is just shorthand notation for i many Xes. Thus, X^i requires space linear in i. Let $\mathcal{A} = (A, \preceq^{\mathcal{A}}, \sqsubseteq^{\mathcal{A}}, c_1, \ldots, c_n)$ be a σ-structure. We evaluate a formula φ on n-dimensional data words $(\overline{a}_i)_{i\geq 1}$ over A where $x_1, \ldots, x_n \in \mathcal{V}$ are the variables occurring in φ. We write a_i^j for the j-th component of $\overline{a}_i$. We say a word $d = (\overline{a}_i)_{i\geq 1}$ *is a model of* φ, denoted as $d \models \varphi$ or $(\overline{a}_i)_{i\geq 1} \models \varphi$, if the following conditions for the atomic comparisons $\sim \in \{=, \preceq, \sqsubseteq\}$ hold:

$$\begin{aligned}
d \models (\mathrm{X}^i\, x_k) \sim (\mathrm{X}^j\, x_\ell) &\iff a_i^k \sim^{\mathcal{A}} a_j^\ell && \text{(i.e., } (a_i^k, a_j^\ell) \in \sim^{\mathcal{A}}\text{)},\\
d \models (\mathrm{X}^i\, x_k) \sim s_j &\iff a_i^k \sim^{\mathcal{A}} s_j^{\mathcal{A}},\\
d \models s_i \sim (\mathrm{X}^j\, x_\ell) &\iff s_i^{\mathcal{A}} \sim^{\mathcal{A}} a_j^\ell,
\end{aligned}$$

where we set $=^{\mathcal{A}}$ as the identity on D^*, and additionally the usual rules for LTL apply:

$$\begin{aligned}
d \models \neg\varphi &\iff (\overline{a}_i)_{i\geq 1} \not\models \varphi,\\
d \models (\varphi_1 \wedge \varphi_2) &\iff (\overline{a}_i)_{i\geq 1} \models \varphi_1 \text{ and } (\overline{a}_i)_{i\geq 1} \models \varphi_2,\\
d \models \mathrm{X}\,\varphi &\iff (\overline{a}_{i+1})_{i\geq 1} \models \varphi,\\
d \models \varphi_1\,\mathrm{U}\,\varphi_2 &\iff \text{there is a } k \in \mathbb{N}_0 \text{ with } (\overline{a}_{i+k})_{i\geq 1} \models \varphi_2\\
&\qquad\qquad \text{and } (\overline{a}_{i+j})_{i\geq 1} \models \varphi_1 \text{ for all } 0 \leq j < k.
\end{aligned}$$

Note that the symbol X has two uses in the logic: either in front of a formula to denote that this formula should hold in the next step, or in front of a variable to denote that the value of this variable at the next step should be considered.

From the logical and temporal connectives defined above, one can derive disjunction, globally, and eventually as usual:

$$\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2), \qquad \mathrm{F}\,\varphi = \top\,\mathrm{U}\,\varphi, \qquad \mathrm{G}\,\varphi = \neg\,\mathrm{F}\,\neg\varphi.$$

Our constraint LTL does not use atomic propositions. On nontrivial structures, proposition p can be resembled by constraints of the form $x_i = c_p$ where we introduced a distinct constant c_p for every proposition p.

Constraint LTL permits arbitrary finite lookahead. In the next proposition, we show that a one-step lookahead suffices.

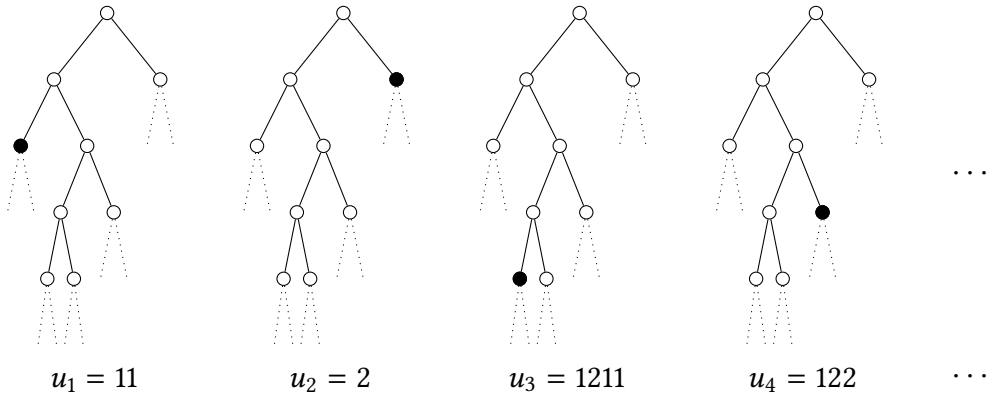


Figure 8.1: 1-dimensional data word $u = (u_i)_{i \geq 1}$ from Example 8.2 (1)

Proposition 8.1. There is a polynomial time algorithm that computes, on input of a cLTL-formula φ, an equivalent cLTL-formula φ' such that φ' does not contain terms of the form $\mathrm{X}^i\, x$ with $i \geq 2$.

Proof. We can replace any occurrence of $(\mathrm{X}^i\, x) \sim (\mathrm{X}^j\, y)$ by $\mathrm{X}^{\min(i,j)}\big((\mathrm{X}^{i-\min(i,j)}) \sim (\mathrm{X}^{j-\min(i,j)}\, y)\big)$. Now assume that there is a subformula of the form $\mathrm{X}^i\, x \sim y$ (the case $x \sim \mathrm{X}^j\, y$ is symmetrical). Introducing fresh variables $y_0, y_1, \ldots, y_{i-1}$ we replace this formula by the formula $x \sim y_i$ and add the conjunct $\mathrm{G}(y_0 = y \wedge \bigwedge_{j=1}^{i} y_j = \mathrm{X}\, y_{j-1})$ to φ. This replacement yields an equivalent formula. Iterating this process for all constraints, we obtain the desired formula ψ. For each atomic comparison, we add at most $|\varphi|$ new variables. Thus, the size of the resulting formula is at most quadratic in the size of φ. ∎

Let us conclude this section by giving two examples that show Constraint LTL at work.

Example 8.2. We give two examples of cLTL formulas and their semantics.

1. Consider the formula $\varphi_1 = \mathrm{G}(\mathrm{X}\, x_1 \sqsubset \mathrm{X}\,\mathrm{X}\, x_1 \sqsubset x_1 \vee x_1 \sqsubset \mathrm{X}\,\mathrm{X}\, x_1 \sqsubset \mathrm{X}\, x_1)$, where $\sqsubset$ denotes the strict lexicographic ordering, i.e., $x \sqsubset y$ if $x \sqsubseteq y$ and $x \neq y$. A data word satisfies this formula if the data value at every position is strictly between the two preceding data values. A concrete example of a data word over $\mathcal{T}_2^C$ that satisfies this property is $(u_i)_{i \geq 1}$ with $u_{2k+1} = (12)^k 11$ and $u_{2k+2} = (12)^k 2$ for every $k \geq 0$. The beginning of this word is depicted in Fig. 8.1. If we consider $\mathcal{T}_{\mathbb{Q}}^C$ as underlying tree, it suffices to choose data values of length 1: the data word $(u_i)_{i \geq 1}$ with $u_i = \frac{(-1)^i}{i}$ also satisfies φ_1. Note that the lexicographic order on data words of length 1 is effectively the natural order $\leq$ on $\mathbb{Q}$.

2. Let $\varphi_2 = \mathrm{G}\big((\mathrm{X}\, x_1 \preceq x_1) \wedge (\mathrm{X}\, x_1 \preceq x_2) \wedge \mathrm{F}(x_1 \not\preceq x_2)\big)$. This formula is not satisfiable, which can be seen as follows: Assume $(\overline{u}_i)_{i\geq 0} \models \varphi_2$. By the first and the second clause, we have $u^1_{i+1} \preceq u^1_i \sqcap u^2_i$ for every $i \geq 0$. Moreover, for any index j with $u^1_j \not\preceq u^2_j$, we have $u^1_j \sqcap u^2_j \prec u^1_j$, and therefore $u^1_{j+1} \prec u^1_j$. As φ_2 asserts the existence of infinitely many such indices j, there has to be an infinite, descending $\prec$-chain in $(u^1_i)_{i\geq 0}$. A contradiction.

8.3 Constraint Automata

In the following, we investigate the satisfiability and model checking problems for Constraint LTL over models with data values in one of the trees $\mathcal{T}^C_D$ for $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}\}$ or $D = \{1, \ldots, k\}$ for some $k \in \mathbb{N}$. We follow closely the automata theoretic approach of Vardi and Wolper [VW94] which provides a reduction of model checking for LTL to the emptiness problem of Büchi automata. In order to deal with the constraints, we use *$\mathcal{T}^C_D$-constraint automata* (cf. [G09]) instead of Büchi automata. Next we recall the definition of constraint automata and state our main result concerning emptiness of constraint automata. We then derive analogous results of Vardi and Wolper's decidability results on LTL for cLTL with constraints over $\mathcal{T}^C_D$. A *$\mathcal{T}^C_D$-constraint automaton* is defined as a usual Büchi automaton but instead of labelling transitions by some letter from a finite alphabet we label them by Boolean combinations of constraints which the current and the next data values have to satisfy in order to execute the transition.

Formally, assume $C = \{c_1, \ldots, c_m\}$ and let $S = \{s_1, \ldots, s_m\}$ be a set of constants symbols. Let B^C_n be the set of all propositional logic formulas with atomic formulas of the form $v \sim v'$ where $v, v' \in \{x_1, \ldots, x_n, y_1, \ldots, y_n\} \cup S$ and $\sim \in \{=, \preceq, \sqsubseteq\}$. Thus, B^C_n contains all quantifier-free MSO formulas over the signature $\{=, \preceq, \sqsubseteq\}$ with variables $\{x_i, y_i \mid i = 1, \ldots, n\} \cup S$. For tuples $\overline{u} = (u_1, \ldots, u_n)$, $\overline{v} = (v_1, \ldots, v_n)$ in $(D^*)^n$ and a formula $\psi \in B^C_n$, we write $(\mathcal{T}^C_D, \overline{u}, \overline{v}) \models \psi$ if ψ evaluates to true when the values $u_1, \ldots, u_n$ are used for $x_1, \ldots, x_n$, the value $v_1, \ldots, v_n$ for $y_1, \ldots, y_n$, and the values of the constants from C for the constant symbols S. In other words $(D^*, (\preceq_D, \sqsubseteq_D), \alpha) \models \psi$, in the sense of Definition 4.3, where α is any assignment with $\alpha(x_i) = u_i$, $\alpha(y_i) = v_i$ and $\alpha(s_j) = c_j$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

Definition 8.3. Let $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}\}$ or $D = \{2, \ldots, k\}$ for some $k \in \mathbb{N}$, and C a finite set of constants. An *n-dimensional $\mathcal{T}^C_D$-constraint automaton* is a quadruple $A = (Q, I, F, \delta)$ where

1. Q is a finite, non-empty set – the set of states,
2. $I \subseteq Q$ – the set of initial states,

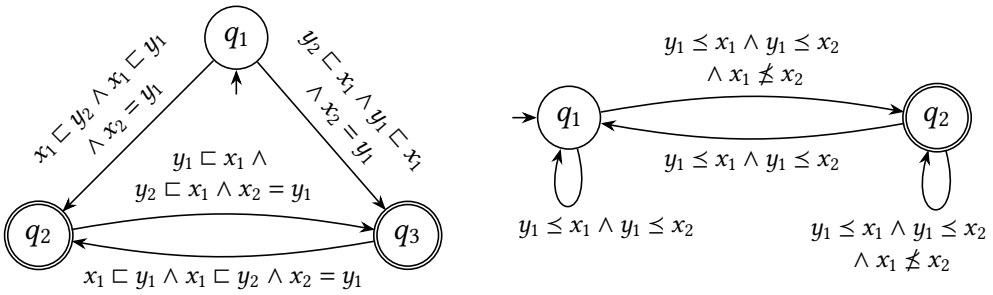


Figure 8.2: Automata described in Example 8.4

3. $F \subseteq Q$ – the set of accepting states,

4. $\delta \subseteq Q \times B_n^C \times Q$ – the *transition relation*,

where B_n^C is defined as above.

A *configuration* of the automaton A is a tuple in $Q \times (D^*)^n$. We define a relation $\rightarrow_A$ on the set of configurations by letting $(q, \overline{u}) \rightarrow_A (p, \overline{v})$ if and only if there is a transition $(q, \beta, p) \in \delta$ such that $(\mathcal{T}_D^C, \overline{u}, \overline{v}) \models \beta$. If A is understood, we just write $\rightarrow$ for $\rightarrow_A$.

A *run* of A is a finite or infinite sequence of configurations $r = (c_j)_{j \in J}$, $J \subseteq \mathbb{N}$ being an interval, such that $c_j \rightarrow c_{j+1}$ for all $j, j+1 \in J$. For a finite run $r = (c_i)_{i_1 \leq i \leq i_2}$ with $i_1 \leq i_2 \in \mathbb{N}$, we say that r is a *run from* c_{i_1} *to* c_{i_2}.

An infinite run $r = (c_i)_{i \in \mathbb{N}}$ is *accepting* if $c_1 = (q, d_1, \ldots, d_n)$ for some initial state $q \in I$ and some final state $f \in F$ appears in infinitely many configurations of r.

The *set of all words accepted by* A consists of all $\overline{w}_1 \overline{w}_2 \cdots \in ((D^*)^n)^\omega$ such that there is an accepting infinite run $(c_i)_{i \in \mathbb{N}}$ with $c_i = (q_i, \overline{w}_i)$.

Example 8.4. We come back to the formulas introduced in Example 8.2:

1. The left automaton in Figure 8.2 depicts an $\mathcal{T}_{\mathbb{Q}}^C$-constraint automaton recognizing the set of data words that model φ_1. Note that φ_1 uses a two-step lookahead, thus we introduced an auxiliary variable x_2, which is always assigned to the next value of x_1. The automaton alternately checks that the current value of x_1 is before the next and next-next value of x_1 or after the next and next-next value of x_1, respectively. An accepting run of A on the data word $\big((11, 2), (2, 1211), (1211, 122), \ldots\big)$ is given below:

$$\begin{array}{ll} Q \rightarrow & \begin{pmatrix} q_1 \\ 11 \\ 2 \end{pmatrix} \begin{pmatrix} q_2 \\ 2 \\ 1211 \end{pmatrix} \begin{pmatrix} q_3 \\ 1211 \\ 122 \end{pmatrix} \begin{pmatrix} q_2 \\ 122 \\ 121211 \end{pmatrix} \begin{pmatrix} q_3 \\ 121211 \\ 12122 \end{pmatrix} \begin{pmatrix} q_2 \\ 12122 \\ 12121211 \end{pmatrix} \begin{pmatrix} q_3 \\ 12121211 \\ 1212122 \end{pmatrix} \cdots . \\ x_1 \rightarrow & \\ x_2 \rightarrow & \end{array}$$

2. The right automaton is built following the intuition of φ_2. At every transition the next value of x_1 must be below the current values of x_1 and x_2 and infinitely

often x_1 must not be a prefix of x_2. By the same reasoning as in Example 8.2, the language of this automaton is empty.

In the following chapter (see Theorem 9.1) we prove that emptiness of n-dimensional $\mathcal{T}_{\mathbb{Q}}^C$-constraint automata is decidable in space linear in $n^K(\log(m) + \log(|C|) + \log(|A|))$ for some global constant K, where m is the length of the longest constant occurring in C. We next apply this result in order to obtain PSPACE-completeness of satisfiability and model checking.

8.4 Satisfiability and Model Checking of Constraint LTL

We define the satisfiability an model checking problem for cLTL over the infinite tree. We prove that these problems for $\mathcal{T}_{\mathbb{Q}}$ are decidable in polynomial space assuming Theorem 9.1. Afterwards, we give a reduction of the the model checking and satisfiability problem for $\mathcal{T}_D$ with $D \neq \mathbb{Q}$ to the case $D = \mathbb{Q}$.

Definition 8.5. Let $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}\}$ or $D = \{2, \dots, k\}$ for some $k \in \mathbb{N}$.

Let SAT($\mathcal{T}_D$) denote the *satisfiability problem* for cLTL over $\mathcal{T}_D^C$: given a set of constants C and a cLTL-formula φ, is there a data word $(\overline{w}_i)_{i\in\mathbb{N}}$ over $\mathcal{T}_D^C$ such that $(\overline{w}_i)_{i\in\mathbb{N}} \models \varphi$?

Let MC($\mathcal{T}_D$) denote the *model checking problem for $\mathcal{T}_D^C$-constraint automata against* cLTL: given constants C, a $\mathcal{T}_D^C$-constraint automaton A and a cLTL-formula φ, is there a data word $(\overline{w}_i)_{i\in\mathbb{N}}$ over $\mathcal{T}_D^C$ accepted by A such that $(\overline{w}_i)_{i\in\mathbb{N}} \models \varphi$?

Theorem 8.6. The problems SAT($\mathcal{T}_{\mathbb{Q}}$) and MC($\mathcal{T}_{\mathbb{Q}}$) are PSPACE-complete.

Our proof of Theorem 8.6 relies on the statement of Theorem 9.1, which is already given below. As the proof of Theorem 9.1 is rather involved, we postpone this proof to Chapter 9.

Theorem 9.1. Let C be a set of constants and A an n-dimensional $\mathcal{T}_{\mathbb{Q}}^C$-constraint automaton. Let furthermore $m = \max\{|c| \mid c \in C\}$. It is decidable in space linear in $n^K(\log(m) + \log(|C|) + \log(|A|))$, for some global constant K independent of C and A, whether $\mathrm{L}(A) \neq \emptyset$.

Proof (of Theorem 8.6). Since there is an automaton accepting all data words, the satisfiability problem reduces to the model checking problem whence it suffices to prove the claim on model checking. Hardness follows directly from the known results for LTL.

Let $C \subseteq \mathbb{Q}^*$ be a finite set of constants, A a $\mathcal{T}_{\mathbb{Q}}^C$-constraint automaton and $\varphi \in$ cLTL. Due to Proposition 8.1 we can assume that all atomic constraints occurring in φ only concern the current and the next data values. Recall that Vardi and Wolper [VW94] provided a translation from LTL to Büchi automata such that the resulting automaton accepts some word if and only if the word is a model of the formula.

This translation lifts to a translation if cLTL over $\mathcal{T}_{\mathbb{Q}}$ to $\mathcal{T}_{\mathbb{Q}}$-constraint automata. We assume that the reader is familiar with the construction given in [VW94]. A description of this construction can also be found in [BK08, Chapter 5]. We outline the construction below. Let φ be a Constraint LTL formula and $x_1, \ldots, x_n$ be all variables occurring in φ. We denote by $\mathrm{cl}(\varphi)$ the closure of φ, i.e., the set of all subformulas and their negations (where $\neg\neg\psi$ and ψ are identified). As in the classical construction we first give a generalised $\mathcal{T}_{\mathbb{Q}}$-constraint automaton, which is transformed to a $\mathcal{T}_{\mathbb{Q}}$-constraint automaton in a second step. Formally, A generalised $\mathcal{T}_{\mathbb{Q}}$-constraint automaton is a quadruple $A = (Q, I, \mathcal{F}, \delta)$, where Q, I, and δ are the same as in Definition 8.3, and $\mathcal{F} \subseteq \mathcal{P}(Q)$. An infinite run is accepting in A, if it starts in an initial state, and visits a state $f \in F$ infinitely often for every $F \in \mathcal{F}$. We define a generalised $\mathcal{T}_{\mathbb{Q}}$-constraint automaton $A = (Q, I, \mathcal{F}, \delta)$ where

$$
\begin{aligned}
Q &= \{M \in \mathrm{cl}(\varphi) \mid M \text{ is maximally consistent}\},\\
I &= \{M \in Q \mid \varphi \in M\},\\
\mathcal{F} &= \left\{F_{\varphi_1,\varphi_2} \;\middle|\; (\varphi_1 \,\mathrm{U}\, \varphi_2) \in \mathrm{cl}(\varphi)\right\},\\
\delta &= \Big\{(M, \beta, M') \;\Big|\; (\mathrm{X}\,\psi) \in M \iff \psi \in M',\\
&\qquad\qquad (\psi_1 \,\mathrm{U}\, \psi_2) \in M \iff \psi_2 \in M' \vee (\psi_1 \,\mathrm{U}\, \psi_2) \in B',\\
&\qquad\qquad \beta = \textstyle\bigwedge_{(v \sim v') \in M} (v \sim v') \wedge \bigwedge_{\neg(v \sim v') \in M} \neg(v \sim v')\Big\},
\end{aligned}
$$

where the v and v' run over all $x_1, \ldots, x_n$, $y_1, \ldots, y_n$ and constants from C, $\sim \in \{=, \preceq, \sqsubseteq\}$, and $F_{\varphi_1,\varphi_2} = \{M \in Q \mid (\varphi_1 \,\mathrm{U}\, \varphi_2) \in M \implies \varphi_2 \in M\}$. Analogous to the classical proof, one shows $(\overline{w}_i)_{i\geq 1} \in \mathrm{L}(A)$ if and only if $(\overline{w}_i)_{i\geq 1} \models \varphi$ for every data word $(\overline{w}_i)_{i\geq 1}$.

Let us outline how to obtain a $\mathcal{T}_{\mathbb{Q}}$-constraint automaton. See [BK08, Chapter 4] for details. Assume $\mathcal{F} = \{F_1, \ldots, F_k\}$. We define the automaton $A' = (Q', I', F', \delta')$ with $Q' = Q \times \{1, \ldots, k\}$, $I' = \{(q, 1) \mid q \in I\}$, $F' = \{(q, 1) \mid q \in F_1\}$ and

$$
\delta' = \left\{\big((q, i), \beta, (q', j)\big) \;\middle|\; (q, \beta, q') \in \delta, j = i + \mathbb{1}_{F_i}(q)\right\},
$$

where we identify $(q, k+1)$ with $(q, 1)$. Note that the number of state of A' is bounded by $|\mathrm{cl}(\varphi)| \cdot |\varphi|$. Thus, the number of states is at most exponential in the size of φ.

Hence, we obtain a constraint automaton A' such that A' accepts $(\overline{w}_i)_{i\in\mathbb{N}}$ if and only if $(\overline{w}_i)_{i\in\mathbb{N}} \models \varphi$. Since the usual product construction for Büchi automata lifts

also to constraint automata, we easily construct an automaton A'' such that A'' accepts a word if and only if both A' and B accept this word. Hence, the set of all words accepted by A'' is non-empty if and only if there is a data word $(\overline{w}_i)_{i\in\mathbb{N}}$ such that A accepts $(\overline{w}_i)_{i\in\mathbb{N}}$ and $(\overline{w}_i)_{i\in\mathbb{N}} \models \varphi$. Since the translation from an LTL formula to a Büchi automaton may result in an exponential size blow-up, we cannot pass this automaton directly to the algorithm checking the emptiness. Instead, using the same idea as in [VW94], whenever the algorithm needs to guess a state or a transition we run a PSPACE decision procedure to verify whether an arbitrarily guessed string of polynomial length is a state or a transition. Furthermore, the size of a single state or a transition is polynomial. Thus, the claim follows. ∎

The rest of this chapter is devoted to showing how $\mathrm{MC}(\mathcal{T}_D^C)$ can be reduced to $\mathrm{MC}(\mathcal{T}_{\mathbb{Q}}^C)$ in logarithmic space. As first step we introduce σ-embeddings which can be used to map runs to different domains.

Definition 8.7. Let σ be a signature, and $\mathcal{A}$ and $\mathcal{B}$ be σ-structures. We say a function $h : \mathcal{A} \to \mathcal{B}$ is a *σ-embedding* if it is injective and preserves the relations, and constants under images and preimages. Formally,

$$\begin{aligned}(a_1, \dots, a_n) \in R^{\mathcal{A}} &\iff (h(a_1), \dots, h(a_n)) \in R^{\mathcal{B}},\\ h(c^{\mathcal{A}}) &= c^{\mathcal{B}},\end{aligned}$$

for all relations $R \in \sigma$, constants $c \in \sigma$ and $a_1, \dots, a_n \in \mathcal{A}$.

We will use the following fact in several places throughout this and the next chapter.

Proposition 8.8. Let $\sigma = \{\preceq, \sqsubseteq, S\}$ and $A = (Q, T, I, F)$ be a $\mathcal{T}_D^C$-constraint automaton and $h\colon D^* \to \mathbb{Q}^*$ a σ-embedding. Then, every finite or infinite sequence $r = (q_i, \overline{w}_i)_{i\in I}$ of configurations is a run in A, if and only if the sequence $h(r) = (q_i, h(\overline{w}_i))_{i\in I}$, where $h(\overline{w}_i) = (h(w_i^j))_{j=1}^n$, is a run in A.

Proof. This is a direct consequence of the fact that h preserves the relations $\preceq, \sqsubset$ and the constants in both directions, and that the transition relation $\to$ of A only depends on these relations and constants. ∎

In the next lemma, we first show that the infinite tree with branching domain $\mathbb{Q}$, can be $\{\preceq, \sqsubseteq\}$-embedded in the infinite binary tree. After this, we give several constructions of σ-embeddings.

Lemma 8.9. Let $\sigma = \{\preceq, \sqsubseteq\}$. There is a σ-embedding from $\mathcal{T}_{\mathbb{Q}}$ to $\mathcal{T}_2$.

Proof. We first show how $(\mathbb{Q}, \leq)$ can be embedded into $(\{1,2\}^*, \sqsubseteq)$ and afterwards extend this mapping to $\mathcal{T}_\mathbb{Q}$.

Let $\mathcal{O} = (\{11, 22\}^*12, \sqsubseteq)$ where $\sqsubseteq$ denotes the lexicographical order. We show that $\mathcal{O}$ and $(\mathbb{Q}, \leq)$ are isomorphic. The domain of $\mathcal{O}$ is countable and does not have endpoints because $(11^n12)_{n\in\mathbb{N}}$ forms a strictly descending sequence such that for any element x of $\mathcal{O}$ there is an $n \geq 0$ with $(11)^n12 \sqsubseteq x$. Analogously, $(22^n12)_{n\in\mathbb{N}}$ is a strictly increasing sequence majorising every element. Thus, it is left to show that $\sqsubseteq$ is a dense order. Assume $w, v \in \mathcal{O}$ with $w \neq v$ and $w \sqsubseteq v$. Let $w = w_1w_2 \dots w_k$ and $v = v_1v_2 \dots v_\ell$ with $w_i, v_i \in \{11, 12, 22\}$. Furthermore, let i be minimal such that $w_i \neq v_i$. If $v_i = 12$ then $w_i = 11$ and $w_1w_2 \dots w_i(22)^{k-i}12$ is between w and v. If $v_i = 22$ then $w_i = 11$ or $w_i = 12$. Hence, $w \sqsubset w_1w_2 \dots w_{i-1}22(11)^{\ell-i}12 \sqsubset v$. Finally, the case $v_i = 11$ is not possible as this would imply $v \sqsubset w$. Thus, $\mathcal{O}$ is a countable, dense order without end points and therefore isomorphic to $(\mathbb{Q}, \leq)$. For the rest of the proof let $h\colon \mathcal{O} \to \mathbb{Q}$ denote such an isomorphism.

We now extend h to a mapping $g\colon \mathbb{Q}^* \to \{1,2\}^*$ by defining

$$g(q_1q_2 \cdots q_n) = h(q_1)\, h(q_2) \cdots h(q_n),$$

i.e., g is the extension of h to $\mathbb{Q}^*$ as a homomorphism. We show that g is a σ-embedding.

We show that g preserves $\leq$ (in both directions). It is obvious from the definition that $w \leq v$ implies $g(w) \leq g(v)$. Now assume that $g(w) \leq g(v)$ and let $w = w_1 \cdots w_k$ and $v = v_1 \cdots v_\ell$. By assumption we have $h(w_1) \cdots h(w_k)w' \leq h(v_1) \cdots h(v_\ell)$. As $\{11, 22\}^*12$ forms a $\leq$-antichain, i.e., all elements are pairwise $\leq$-incomparable, any word u in $(\{11, 22\}^*12)^*$, can be uniquely decomposed into words $u = u_1 \cdots u_\ell$ with $u_i \in \{11, 22\}^*12$ for all $i = 1, \dots, \ell$, c.f. Proposition 6.30. Hence, we conclude $k \leq \ell$ and $h(w_i) = h(v_i)$ for all $i = 1, \dots, k$. By injectivity of h, we obtain $w \leq v$. Note that this also shows the injectivity of g, as $\leq$ is a partial order.

We prove preservation of $\sqsubseteq$. Let $w = w_1 \cdots w_k$ and $v = v_1 \cdots v_\ell$. Assume $w \sqsubseteq v$. If $w \leq v$, we have $g(w) \leq g(v)$ by the previous paragraph. Thus, assume $w_i < v_i$ and $w_j = v_j$ for a $i \leq \min(k, \ell)$ and all $j < i$. Since $w_i < v_j$ implies $h(w_i) \sqsubset h(v_i)$, we conclude $g(w) \sqsubset g(v)$. Conversely, assume $g(w) \sqsubseteq g(v)$. If $g(w) \leq g(v)$ we conclude $w \leq v$ by the previous paragraph. Let i be minimal with $h(w_i) \neq h(v_i)$. Since $h(w) \sqsubset h(v)$, we obtain $h(w_i) \sqsubset h(v_i)$. As h is an isomorphism, we conclude $w_i < v_i$ and $w_j = v_j$ for all $j < i$. Therefore, $w \sqsubset v$. ∎

We are now ready to give the reduction from the model checking problem over $\mathcal{T}_D$ to the model checking problem over $\mathcal{T}_\mathbb{Q}$.

Lemma 8.10. Let $D \in \{\mathbb{N}, \mathbb{Z}\}$ or $D = \{1, \dots, k\}$ for some $k \geq 2$. Then, $\mathrm{MC}(\mathcal{T}_D)$ is LOGSPACE-reducible to $\mathrm{MC}(\mathcal{T}_\mathbb{Q})$.

Proof. Let $C \subseteq D^*$ be a finite set of constants, A a $\mathcal{T}_D^C$-constraint automaton and $\varphi \in$ cLTL. Assume variables $x_1, \ldots, x_n$ occur in φ. We may assume, that C is closed under prefixes, as any set can be closed under prefixes with only polynomial blowup. The crucial difference to the case $D = \mathbb{Q}$ is that the branching domain is not dense and possibly bounded. For every $i \in \{1, \ldots, n\}$, we define the following formulas:

$$\alpha_i = \bigwedge_{\substack{c \in C, j \in D: \\ cj, c(j+1) \in C}} \big(cj \sqsubseteq x_i \implies (cj \preceq x_i \vee c(j+1) \sqsubseteq x_i)\big),$$

$$\beta_i^0 = \begin{cases} \bigwedge_{c \in C} (c1 \sqsubseteq x \vee x \preceq c1) & \text{if } D \neq \mathbb{Z} \\ \top & \text{otherwise,} \end{cases}$$

$$\beta_i^1 = \begin{cases} \bigwedge_{c \in C} (x \sqsubseteq ck \vee ck \preceq x) & \text{if } D = \{2, \ldots, k\} \\ \top & \text{otherwise,} \end{cases}$$

where $\top$ is some fixed tautology. We claim that (C, A, φ) is a positive instance of $\mathrm{MC}(\mathcal{T}_k^C)$ if and only if (C', A, ψ) is a positive instance of $\mathrm{MC}(\mathcal{T}_{\mathbb{Q}}^C)$, where $C' \supseteq C$ additionally contains the constants used in $\beta_i^{0,1}$, A is seen as a $\mathcal{T}_{\mathbb{Q}}^C$-automaton, and $\psi = \varphi \wedge \mathrm{G} \bigwedge_{i=1}^n (\alpha_i \wedge \beta_i^0 \wedge \beta_i^1)$. Intuitively, ψ is obtained from φ by adding checks that the data values may not occur between constants of the form cj and $c(j+1)$, and not before the minimal possible data value or after the maximal possible data value. Thus, we need to show that there is a witness for the instance (C, A, φ) if and only if there is a witness for (C', A, ψ), i.e., that the following conditions are equivalent:

1. $d \models \varphi$ for some $d \in \mathrm{L}_D(A)$,
2. $d \models \psi$ for some $d \in \mathrm{L}_{\mathbb{Q}}(A)$.

First, assume statement 1 holds. Let $d = (\overline{u}_i)_{i \geq 0}$ with $d \in \mathrm{L}_D(A)$ and $d \models \varphi$. As every u_i^j is in D^*, we automatically obtain $d \models \mathrm{G}(\alpha_n \wedge \beta_0^n \wedge \beta_1^n)$. Moreover, note that $\models$ does not depend on the tree arity. So $d \models \varphi$ regardless of whether we consider φ as a formula over D or over $\mathbb{Q}$. Furthermore, $d \in \mathrm{L}_D(A)$ implies $d \in \mathrm{L}_{\mathbb{Q}}(A)$ as the automaton only depends on the relations between data values and not on the tree domain. Together we obtain statement 2.

The converse direction is more involved. We construct a $\{\preceq, \sqsubseteq\}$-embedding h which maps a witness of statement 2 to a witness of statement 1. In order to define this function, we first need to define its domain: let $K \subseteq \mathbb{Q}^*$ contain all words w such that

1. $ci \sqsubseteq w \implies (ci \preceq w \vee c(i+1) \sqsubseteq w)$ for all $c \in D^*$, $i \in \mathbb{Q}$ with $ci, c(i+1) \in C$.

2. if $D = \mathbb{N}$ or $D = \{1, \dots, k\}$, then $c1 \sqsubseteq w$ for all $c \in C$

3. if $D = \{1, \dots, k\}$, then $w \sqsubseteq ck$ or $ck \preceq w$ for all $c \in C$.

Clearly, $(\overline{u}_i)_{i \geq 0} \models \mathrm{G} \bigwedge_{i=1}^{n} (\alpha_i \wedge \beta_0^i \wedge \beta_1^i)$ if and only if $u_i^j \in K$ for all $i \geq 0$ and $j \in \{1, \dots, n\}$. We define a mapping $\{\preceq, \sqsubseteq\}$-embedding $h \colon K \to D^*$. Intuitively, h maps all nodes of the form cq for some constant c below the node $c(j+1)$, where $j < q$ is maximal with $cj \in C$. By the choice of K, $c(j+1)$ cannot be a constant.

Let $z_{\min}$ be 1 if $D = \mathbb{N}$ or $D = \{1, \dots, k\}$ or smaller than any component of any constant if $D = \mathbb{Z}$. For every $c \in C$ let the function $\iota_c \colon \mathbb{Q} \to D$ be given by

$$\iota_c(q) = \begin{cases} \max\{z \leq q \mid cz \in C\} + 1 & \text{if there is a } z \leq q \text{ with } cz \in C \\ z_{\min} & \text{otherwise.} \end{cases}$$

Let $g \colon \mathbb{Q}^* \to \{1, 2\}^*$ be a $\{\preceq, \sqsubseteq\}$-embedding. With the help of ι_c we define a function $h \colon K \to D^*$ by

$$h(w) = \begin{cases} w & \text{if } w \in C \\ c\, \iota_c(q)\, g(qu) & \text{if } c \in C \text{ is maximal with } c \preceq w \text{ and } w = cqu \\ & \qquad \text{for some } q \in D, u \in D^*. \end{cases}$$

This mapping is a $\{\preceq, \sqsubseteq, S\}$-embedding. The rather technical proof of this statement is outsourced to Lemma 8.11.

Now, assume $d \models \varphi \wedge \mathrm{G}(\alpha^n \wedge \beta_0^n \wedge \beta_1^n)$ for some $d \in \mathrm{L}_{\mathbb{Q}}(A)$. Let $d = (\overline{u}_i)_{i \geq 0}$. Thus, we have $u_i^j \in K$ for all i and j. Hence, we can apply h to the data word, obtaining a data word $h(d) = (h(\overline{u}_i))_{i \geq 0}$ with $h(\overline{u}_i) = (h(u_i^1), \dots, h(u_i^n))$. As h is a $\{\preceq, \sqsubseteq, S\}$-embedding, we obtain $h(d) \models \varphi$ and $h(d) \in \mathrm{L}_{\mathbb{Q}}(A)$ by Proposition 8.8. As $h(d) \in ((D^*)^n)^\omega$, we also have $h(d) \in \mathrm{L}_D(A)$. This completes the proof. ∎

Lemma 8.11. We assume the notation of the proof of Lemma 8.10. The mapping $h \colon K \to D^*$ is a $\{\preceq, \sqsubseteq, S\}$-embedding.

Proof. Recall that we assume the set of constants to be closed under prefixes. We show that h is well-defined. The only case that may violate well-definedness of h is $D = \{1, \dots, k\}$ that $\iota_c(q) > k$. Let $w = cqx$ with $c \preceq w$ and $c \in C$ maximal. Since $w \in K$, we conclude $q \leq k$. As cq is not a constant, $\iota_c(q) \leq k$ and h is well-defined.

We prove the following statement for all $w \in K$: if $w \notin C$, then $h(w) \notin C$. Let $w = cqw'$ with $c \preceq w$ maximal and $q \in \mathbb{Q}$. Consider the case that there is no $z \in D$ with $z \leq q$ and $cz \in C$. If $D = \mathbb{Z}$, we have $c\, \iota_c(q) \notin C$ by construction of ι_c. If $D = \mathbb{N}$ or $D = \{1, \dots, k\}$, we conclude $c1 \sqsubseteq w$ as $w \in K$ and so $1 \leq q$. Since we assumed that there is no constant cz with $z \leq q$, $c1 = c\, \iota_c(q)$ is not a constant. As C is closed

under prefixes $h(w)$ is also not a constant. In the case that there is a $z \in D$ with $cz \in C$ and $z \leq q$, let z_0 be maximal with this property. Since $cq \in K$ and $cz_0 \sqsubseteq cq$, we have $z_0 = q$ or $z_0 + 1 \leq q$. As $cq \notin C$, only the case $z_0 + 1 \leq q$ can occur. Thus, by maximality of z_0, $c(z_0 + 1) = c\,\iota_c(q)$ is not a constant. Therefore, $h(w) \notin C$.

Preservation of $\preceq$: We show that h preserves $\preceq$ in both directions. Let $u, v \in K$. Assume $v \in C$. Then, $h(v) = v$. If $u \preceq v$, then u is also a constant, and $h(u) = u$. If $h(u) \preceq h(v) = v$, we conclude that $h(u)$ is a constant. By the second paragraph, u is a constant, and $u = h(u) \preceq h(v) = v$.

Next, we consider the case that $u \in C$ and $v \notin C$. If $u \preceq v$, there is a constant c with $u \preceq c \preceq v$. By definition of h, we obtain $h(u) = h(c) \preceq h(v)$. Conversely, assume $h(u) \preceq h(v)$. As $h(v)$ is not a constant, there is a maximal constant $c \in C$ with $h(u) = u \preceq c \prec h(v)$. Assume $c \not\preceq v$. Let c' be maximal with $c' \preceq v$. Thus, $c' \preceq h(v)$. By maximality of c we conclude $c' \preceq c$. By definition of h and the second paragraph, c' is the maximal constant that is a prefix of $h(v)$. Thus, $c = c'$ and $u \preceq v$.

Assume $u, v \notin C$. Let $u = cqx$ and $v = c'q'x'$, where $c, c \in C'$ are maximal with $c \preceq u$ and $c' \preceq v$. If $u \preceq v$, then $c = c'$, $q = q'$, and $x \preceq x'$ since $cq, c'q' \notin C$. We conclude $h(u) = c\,\iota_c(q)\,g(qx) \preceq c\,\iota_c(q)\,g(q'x') = h(v)$. Conversely, if $h(u) \preceq h(v)$, we have $c\,\iota_c(q)\,g(qx) \preceq c'\,\iota_c(q')\,g(q'x')$. As before, we conclude $c = c'$, $\iota_c(q) = \iota_{c'}(q')$, and $g(qx) \preceq (q'x')$. As g preserves $\preceq$, this implies $qx \preceq q'x'$, i.e., $q = q'$ and $x \preceq x'$. Therefore, $u \preceq v$.

Preservation of $\sqsubseteq$: We show that h preserves $\sqsubseteq$ in both directions. Let $u, v \in K$ with $u \sqsubseteq v$. If $u \preceq v$ holds, we conclude $h(u) \sqsubseteq h(v)$, as h preserves $\preceq$. Assume $u = wqx$ and $v = wq'x'$ with $q < q'$. If wq and wq' are constants, we obtain $h(u) \sqsubseteq h(v)$ by definition. Assume that only wq is a constant. Thus, $\iota_c(q') > q$ and $wq \sqsubseteq w\,\iota_c(q')$. This implies $h(u) \sqsubseteq h(v)$. Next, assume only wq' is a constant. Assume there is a $z \leq q$ with $cz \in C$. Since $q < q'$, we have $z < q'$. Thus, $\iota_c(q) = z + 1 \leq q'$. If there is no such z, then $\iota_c(q) \leq q'$ by definition of ι_c. In both cases, we obtain $\iota_c(q) < q'$ since $w\iota_c(q)$ is not a constant by the second paragraph. We conclude $h(u) \sqsubseteq h(v)$ as g preserves $\sqsubseteq$. Finally, assume $wq, wq' \notin C$, but $w \in C$. As ι_c is monotonic, we obtain $\iota_w(q) \leq \iota_c(q')$ and therefore $h(u) = w\,\iota_w(q)\,g(qx) \sqsubseteq w\,\iota_w(q')\,g(q'x') = h(v)$. If $w \notin C$, we conclude $h(u) = h(w)g(qx)$ and $h(v) = h(w)g(q'x')$. As g preserves $\sqsubseteq$, we obtain $h(u) \sqsubseteq h(v)$.

Conversely, assume $u, v \in K$ with $h(u) \sqsubseteq h(v)$. If $h(u) \preceq h(v)$ we immediately conclude $u \sqsubseteq v$. Moreover, if u and v are constants, the claim follows immediately. Assume $u \in C$, $v = cq'x'$ with $c \in C$ maximal with $c \preceq v$. Since $u = h(u) \sqsubseteq h(v) = c\,\iota_c(q')\,g(x')$ and $c\,\iota_c(q) \notin C$, the maximal common prefix of u and $h(v)$ is a strict prefix of u and c. Thus, $u \sqsubset c$. The case that only v is a constant is analogous. We still need to consider the case that both words u,v are not constants. Let $u = cqx$ and $v = c'q'x'$. Assume $c = c'$. Then, either $\iota_c(q) = \iota_c(q')$ and $g(qx) \sqsubseteq g(q'x')$, or $\iota_c(q) < \iota_c(q')$ and $q < q'$. In both cases we conclude $u \sqsubseteq v$. If $c \prec c'$, we have

$h(cq) \sqsubset h(c')$ and conclude $cq \sqsubset c'$ as above. Analogously, if $c' \prec c$, we obtain $c \sqsubset c'q'$ from $h(c) \sqsubset h(c'q')$. Thus, assume that c and c' are $\preceq$-incomparable. Hence, $c \sqsubset c'$ and so $u \sqsubset v$.

Injectivity follows from the fact that h preserves the partial order $\preceq$. This completes the proof. ∎

From Theorem 8.6 and Lemma 8.10 we directly obtain the following corollary.

Corollary 8.12. Let $D \in \{\mathbb{Q}, \mathbb{N}, \mathbb{Z}\}$ or $D = \{1, \ldots, k\}$ for some $k \geq 2$. Then, $\mathrm{MC}(\mathcal{T}_D^C)$ and $\mathrm{SAT}(\mathcal{T}_D^C)$ are PSPACE-complete.

Remark 8.13. Demri and Deter [DD15] conjectured that if the arity k of the tree is part of the input to the satisfiability problem, it is still in PSPACE. Our proof confirms that this branching degree uniform satisfiability problem is PSPACE-complete.

Chapter 9

Emptiness of Tree Constraint Automata

Recall that every non-empty Büchi automaton has an accepting run which is ultimately periodic. We first prove that a nonempty constraint automaton has an accepting run which ultimately consists of loops that never contract the distances of data values and keep the order type of the data values constant. We then define the notion of the type of a run. It turns out that such a non-contracting loop exists if and only if the automaton has a run realising a type among a certain set. Finally, we provide a nondeterministic algorithm, which uses space polynomial in the dimension of the automaton, but logarithmic in the automaton's size, that checks whether an automaton realises a given type. Putting all these together yields our main technical result:

Theorem 9.1. Let C be a set of constants and A an n-dimensional $\mathcal{T}_{\mathbb{Q}}^{C}$-constraint automaton. Let furthermore $m = \max\{|c| \mid c \in C\}$. It is decidable in space linear in $n^K(\log(m) + \log(|C|) + \log(|A|))$, for some global constant K independent of C and A, whether $\mathrm{L}(A) \neq \emptyset$.

The proof of this theorem will take up the rest of this chapter.

This is joint work with Alexander Kartzow. The results can also be found in [KW15].

9.1 Emptiness and Stretching Loops

We first introduce some notation before defining our notion of stretching loop and characterising emptiness in terms of stretching loops.

From now on a *word* is always an element of $\mathbb{Q}^*$, $\bigsqcap$ ($\sqcap$) denotes the (binary) *greatest common prefix operator*, and we fix a finite tuple of words $C = (c_1, c_2, \ldots, c_m)$ called constants. Moreover, we fix a $\mathcal{T}_{\mathbb{Q}}^{C}$-constraint automaton A with state space Q.

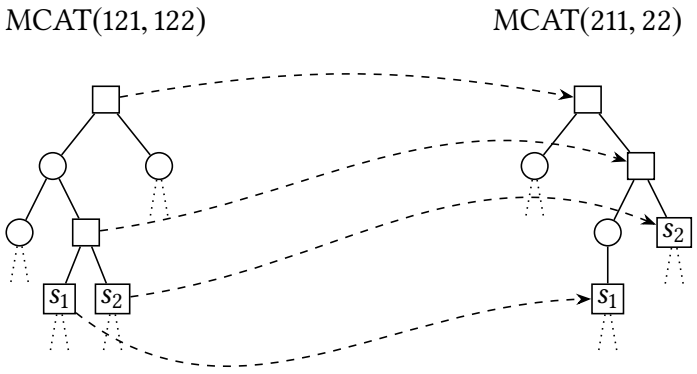


Figure 9.1: Situation described in Example 9.3. The dotted arrows represent the isomorphism h: $\mathrm{MCAT}(121, 122) \to \mathrm{MCAT}(211, 22)$

We assume that C is closed under prefixes.

Note that a reference to the prefix of a constant can be stored in space logarithmic in the number of constants and the maximal length of the constants, by storing the index of the constant and the length of the prefix. Thus, this assumption does not increase the space needed by our algorithm as the algorithm never stores the actual value of a constant, but merely references the constants.

Definition 9.2. Let $s_1, s_2, \ldots$ be countable many constant symbols. Given a tuple $\overline{w} = (w_1, w_2, \ldots, w_n)$ of words, the *maximal common ancestor tree* $\mathrm{MCAT}(\overline{w})$ of $\overline{w}$ is the following σ-structure, where $\sigma = \{\preceq, \sqsubseteq, s_1, s_2, \ldots, s_n\}$:

$$\begin{aligned}\mathrm{MCAT}(\overline{w}) = (M, \preceq|_{M^2}, \sqsubseteq|_{M^2}, w_1, w_2, \ldots, w_n) \text{ with}\\ M = \{\varepsilon\} \cup \{\textstyle\bigsqcap_{i\in I} w_i \mid \emptyset \neq I \subseteq \{1, 2, \ldots, n\}\},\end{aligned}$$

i.e., w_i is the interpretation of constant symbol s_i.

The *(order) type* $\mathrm{typ}(\overline{w})$ *of* $\overline{w}$ is the σ-isomorphism class of $\mathrm{MCAT}(\overline{w})$. We define $\mathrm{MCAT}_C(\overline{w}) = \mathrm{MCAT}(\overline{w}, C)$ and $\mathrm{typ}_C(\overline{w}) = \mathrm{typ}(\overline{w}, C)$, i.e., $\mathrm{MCAT}_C(\overline{w})$ includes all positions from $\overline{w}$ as well as all positions from C.

Labelling the words from $\overline{w}$ by constant symbols has the following consequence: if $\mathrm{typ}_C(\overline{w}) = \mathrm{typ}_C(\overline{v})$ for $\overline{w} = (w_1, w_2, \ldots, w_n)$ and $\overline{v} = (v_1, v_2, \ldots, v_n)$ then there is a unique isomorphism h from $\mathrm{MCAT}_C(\overline{w})$ to $\mathrm{MCAT}_C(\overline{v})$ which maps $c \mapsto c$ for every $c \in C$ and $w_i \to v_i$ for w_i the i-th element of $\overline{w}$ and v_i the i-th element of $\overline{v}$.

Example 9.3. We consider the 2-dimensional data values $\overline{u} = (121, 122)$ and $\overline{v} = (211, 22)$ in $\mathcal{T}_2$. In Fig. 9.1 both tuples are shown embedded in the binary tree. The rectangular nodes correspond to the maximal common ancestor tree of $\overline{u}$, $\overline{v}$, respectively. Clearly, we have $\mathrm{typ}(\overline{u}) = \mathrm{typ}(\overline{v})$. Thus, there exists a unique σ-isomorphism $h\colon \mathrm{MCAT}(\overline{u}) \to \mathrm{MCAT}(\overline{v})$.

We introduce a partial order $\leq_C$ on the set of configurations.

Definition 9.4. We make the following definitions:

1. Let $D \subseteq \mathbb{Q}^*$. A function $h\colon D \to \mathbb{Q}^*$ is called *stretching* if $|h(e)|-|h(d)| \geq |e|-|d|$ for all $d, e \in D$ with $d \preceq e$.

2. For $n \in \mathbb{N}$ we define a relation $\leq_C$ on configurations from $Q \times (\mathbb{Q}^*)^n$ by $(q, \overline{w}) \leq_C (p, \overline{v})$ if $q = p$, $\mathrm{typ}_C(\overline{w}) = \mathrm{typ}_C(\overline{v})$ and the induced isomorphism $h : \mathrm{MCAT}_C(\overline{w}) \to \mathrm{MCAT}_C(\overline{v})$ is stretching.

Intuitively, $(q, \overline{w}) \leq_C (q, \overline{v})$ holds if both data tuples have the same order type and the distances between parent nodes and direct child nodes in $\mathrm{MCAT}_C(\overline{v})$, seen as a subtree of $\mathbb{Q}^*$, are greater than the corresponding distances in $\mathrm{MCAT}_C(\overline{w})$. Note that the isomorphism h shown in Fig. 9.1 is not stretching since $|12| - |\varepsilon| = 2 > 1 = |1| - |\varepsilon| = |h(12)| - |h(\varepsilon)|$.

Recall that a well-quasi ordering is a quasi ordering R, i.e., R is reflexive and transitive, such that for any infinite sequence $x_1, x_2, \ldots$ of elements there are indices $i < j$ with $(x_i, x_j) \in R$.

Lemma 9.5. $\leq_C$ is a well-quasi order.

Proof. Obviously, $\leq_C$ is a quasi order.

Let $(\overline{w}_i)_{i\in\mathbb{N}}$ be an infinite sequence of n-tuples of words. This sequence induces an infinite subsequence $(\overline{w}_i, C)_{i\in I}$ such that for all $i, j \in I$ $\mathrm{typ}_C(\overline{w}_i) = \mathrm{typ}_C(\overline{w}_j)$. This implies that $\mathrm{MCAT}_C(\overline{w}_i)$ and $\mathrm{MCAT}_C(\overline{w}_j)$ are isomorphic for all $i, j \in I$ via an isomorphism $\phi_{i,j}$.

For every $i \in I$ we define a map $f_i : \mathrm{MCAT}_C(\overline{w}_i)^2 \to \mathbb{N}$ by $(u, v) \mapsto |u| - |u \sqcap v|$. Fix an $i_0 \in I$ and an enumeration of the domain of f_{i_0}. This induces an enumeration of the domain of f_i for every $i \in I$ by letting $(u, v) \in \mathrm{dom}(f_i)$ be the k-th element if $(\phi_{i,i_0}(u), \phi_{i,i_0}(v))$ is the k-th element of $\mathrm{dom}(f_{i_0})$.

Consider the set $\{f(\overline{w}_i) \mid i \in I\} \subseteq \mathbb{N}^n$. By Dickson's Lemma we find tuples $\overline{w}_j$, $\overline{w}_k$ $(j < k)$ such that $f_k(\phi_{j,k}(u), \phi_{j,k}(v)) \geq f_j(u, v)$ for all $(u, v) \in \mathrm{MCAT}_C(\overline{w}_j)$. From this we immediately conclude that $\overline{w}_j \leq_C \overline{w}_k$. ∎

We want to show that the order $\leq_C$ and the relation $\to$ induced by the transitions of a constraint automaton are compatible in the sense of strong upwards compatibility. We say that $\to$ is *strongly upwards compatible* with respect to $\leq_C$ if for all configurations (q, u), (p, v), and (q, u') with $(q, u) \to (p, v)$ and $(q, u) \leq_C (q, u')$ there is a configuration (p, v') such that $(q, u') \to (p, v')$ and $(p, v) \leq_C (p, v')$.

We prepare the proof of strong upwards compatibility of the transition relation by formally proving the following intuition: if $\mathrm{MCAT}_C(\overline{w}')$ has larger gaps than $\mathrm{MCAT}_C(\overline{w})$ (seen as subtrees of $\mathbb{Q}^*$), every extension of $\mathrm{MCAT}_C(\overline{w})$ to a bigger tree induces a corresponding extension of $\mathrm{MCAT}_C(\overline{w}')$ to a bigger tree of the same order type. The proof of this statement requires the following technical lemma, which gives constructions of σ-embeddings.

Lemma 9.6. Let $\sigma = \{\preceq, \sqsubseteq, \sqcap\}$. The following functions are σ-embeddings:

1. For any $u \in \mathbb{Q}^*$ the function $\iota_u \colon \mathbb{Q}^* \to \mathbb{Q}^*$ given by $\iota_u(w) = uw$.
2. For any strictly monotonically increasing, bijective function $\ell \colon \mathbb{Q} \to \mathbb{Q}$, the function $\widetilde{\ell} \colon \mathbb{Q}^* \to \mathbb{Q}^*$ defined by $\widetilde{\ell}(\varepsilon) = \varepsilon$ and $\widetilde{\ell}(q_1 \cdots q_n) = \ell(q_1)q_2 \cdots q_n$.
3. Given two σ-embeddings $f, g \colon \mathbb{Q}^* \to \mathbb{Q}^*$ and a position $z \in \mathbb{Q}^*$, the function $h = f[z \leftarrow g]$ given by
$$h(w) = \begin{cases} f(z)\, g(w') & \text{if } w = zw' \\ f(w) & \text{otherwise.} \end{cases}$$
Moreover, if f, g are only $\{\preceq, \sqsubseteq\}$-embeddings, so is $f[z \leftarrow g]$.
4. Let $\sigma = \{\preceq, \sqsubseteq\}$ or $\sigma = \{\preceq, \sqsubseteq, \sqcap\}$. Given an infinite sequence of σ-embedding $(f_i)_{i \in \mathbb{N}}$ such that for every $x \in \mathbb{Q}^*$ there is an $N \in \mathbb{N}$ with $f_i(x) = f_j(x)$ for all $i, j \geq N$. Then, the function $f \colon \mathbb{Q}^* \to \mathbb{Q}^*$ given by $f(x) = y$ if $f_i(x) = y$ for almost all $i \in \mathbb{N}$ is a σ-embedding.

Proof. **1.** The statement about ι_u follows directly from the definitions of the relations $\preceq$, $\sqsubseteq$, and $\sqcap$.

2. Let $\ell \colon \mathbb{Q} \to \mathbb{Q}$ a strictly monotonic function. Let $u, v \in \mathbb{Q}^*$ with $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_m$. If $u \preceq v$, then $v = uv'$ for some $v' \in \mathbb{Q}^*$. This implies $\widetilde{\ell}(u) = \ell(u_1)u_2 \cdots u_n \preceq \ell(u_1)u_2 \cdots u_n v' = \widetilde{\ell}(v)$. Conversely, assume $\widetilde{\ell}(u) \preceq \widetilde{\ell}(v)$. Thus, $n \leq m$ and $\ell(u_1) = \ell(v_1)$, $u_2 = v_2, \ldots, u_n = v_n$. As ℓ is injective, we conclude $u \preceq v$.

Assume, $u \sqsubseteq v$. If $u \preceq v$, we conclude $\widetilde{\ell}(u) \preceq \widetilde{\ell}(v)$. Thus, assume $u = xqy$ and $v = xq'y'$ with $q, q' \in Q$ and $q < q'$. If x is non-empty, we directly conclude $\widetilde{\ell}(u) \sqsubseteq \widetilde{\ell}(v)$ by definition of $\widetilde{\ell}$. Otherwise, $x = \varepsilon$ and $\widetilde{\ell}(u) = \ell(q)y \sqsubseteq \ell(q')y' = \widetilde{\ell}(v)$ since ℓ is strictly monotonic. Conversely, assume $\widetilde{\ell}(u) \sqsubseteq \widetilde{\ell}(v)$. As before, the case

$\widetilde{\ell}(u) \preceq \widetilde{\ell}(v)$ follows from the previous paragraph. Thus $\widetilde{\ell}(u) = xqy$ and $\widetilde{\ell}(v) = xq'y'$ with $q < q'$. If x is non-empty, we have $u = x_0qy$ and $v = x_0q'y'$ for some $x_0 \in \mathbb{Q}^*$ and so $u \sqsubseteq v$. If $x = \emptyset$, we conclude $q = \ell(q_0)$ and $q' = \ell(q_0')$ for some $q_0, q_0' \in \mathbb{Q}$. As ℓ is strictly monotonic, this implies $q_0 < q_0'$. Hence, $u \sqsubseteq v$.

Note that $\widetilde{\ell}$ is bijective as ℓ is bijective. Thus, $\widetilde{\ell}$ preserves $\sqcap$ as it preserves $\preceq$ in both directions.

3. Let f, g, z as in the assumptions of the statement. Let $u, v \in \mathbb{Q}^*$. If u and v are either both suffixes of z or both strict prefixes of or incomparable to z, then the function h is essentially only the function f or the function g applied to both values. Thus, the statement follows directly by the assumptions on f and g.

Assume u is a strict prefix of z or incomparable to z and $v = zv'$. If $u \preceq v$, we conclude $h(u) = f(u) \preceq f(z) \preceq f(z)\,g(v') = h(v)$. Conversely, assume $h(u) \preceq h(v)$. Thus, $h(u) = f(u)$ is either a strict prefix or a suffix of $f(z)$. If $f(z) \preceq f(u)$, then $z \preceq u$ since f preserves $\preceq$ in both directions. This contradicts the assumptions on u. Hence, $u \prec z \preceq v$. Next, we consider the case $u \sqsubseteq v$, i.e., $u = xqy$ and $v = xq'y'$ with $q < q'$. Since $x \preceq u \sqcap v$, we have $x \prec z$. Thus, $xq' \preceq z$ and $h(xq') \preceq h(v)$. Since $xq \sqsubset xq'$ are $\preceq$-incomparable, the same holds for $f(xq)$ and $f(xq')$. We conclude $h(u) \sqsubset h(v)$ since $f(xq) \preceq h(u)$ and $f(xq') \preceq h(v)$. Conversely, assume $h(u) \sqsubseteq h(v)$ and $h(u) \not\preceq h(v)$. Hence, $h(u) = xqy$ and $h(v) = xq'y'$ with $q < q'$. Since u is not a suffix of z, $f(u)$ is not a suffix of $f(z)$. Hence, x is a strict prefix of $f(z)$. Therefore, $xq' \preceq f(z)$ and $f(u) \sqsubset f(z)$. Since xq' is also a prefix of $f(z)$, we obtain that $f(u)$ and $f(z)$, and u and z are $\preceq$-incomparable We conclude $u \sqsubset z$ and $u \sqsubset v$ as $z \preceq v$ and u and z are $\preceq$-incomparable. Finally, we show that h preserves $\sqcap$. Note that since, u is not a suffix of z, we have $u \sqcap v = u \sqcap z \prec z$. We conclude $h(u \sqcap v) = h(u \sqcap z) = f(u \sqcap z) = f(u) \sqcap f(z) = h(u) \sqcap h(v)$, since $f(u)$ is not a suffix of $f(z)$.

The case that u is a suffix of z and v is not a suffix of z is analogous to the previous case.

4. Consider two words $u, v \in \mathbb{Q}^*$. By definition of f, there is a $N \geq 0$ such that $f(u) = f_N(u)$ and $f(v) = f_N(v)$. As f_N is a σ-embedding, the claim follows. ∎

Lemma 9.7. Let $\sigma = \{\preceq, \sqsubseteq, \sqcap\}$. Let further be $\{\varepsilon\} \subseteq A \subseteq \mathbb{Q}^*$ finite and closed under greatest common prefixes, and f a stretching σ-embedding on A. Then, f extends to a stretching σ-embedding $g\colon \mathbb{Q}^* \to \mathbb{Q}^*$.

Proof. We use induction on $|A|$. For $A = \{\varepsilon\}$, the mapping g is just the identity on $\mathbb{Q}^*$.

Assume $A \supsetneq \{\varepsilon\}$. We show the case $|A| = 2$ separately. Let $x = x_1 \cdots x_m \in A$ and $f(x) = x' = x_1' \cdots x_{m'}'$ with $m' \geq m$ as f is stretching. Fix strictly monotonic,

bijective functions $\ell_i \colon \mathbb{Q} \to \mathbb{Q}$ for $1 \leq i \leq m$ with $\ell_i(x_i) = x_i'$ for $i \leq m$. Define g by

$$g(y_1 \cdots y_n) = \begin{cases} f(x) y_{m+1} \cdots y_n & \text{if } x \preceq y_1 \cdots y_n, \\ \ell_1(y_1) \cdots \ell_n(y_n) & \text{if } y_1 \cdots y_n \prec x \\ \ell_1(y_1) \cdots \ell_k(y_k) y_{k+1} \cdots y_n & \text{otherwise,} \end{cases}$$

where $k \leq n$ is minimal with $y_1 \cdots y_k \not\preceq x$. Obviously, $g(x) = f(x)$. The function g can be written using the operations and functions defined in Lemma 9.6 as follows:

$$g = \mathrm{id}[\varepsilon \leftarrow \ell_1][x_1 \leftarrow \ell_2][x_1 x_2 \leftarrow \ell_3] \cdots [x_1 \cdots x_{m-1} \leftarrow \ell_m][x_1 \cdots x_m \leftarrow \iota_{x'_{m+1} \cdots x'_{m'}}].$$

By the results of the same lemma, g is a σ-embedding.

Next, assume $|A| > 2$. Choose a position $y \in A$ such that the set $X = \{x \in A \mid y \leq x,\ y \neq x\}$ is non-empty and contains only $\preceq$-incomparable elements. Let $A_0 = A \setminus X$ and $f_0 = f|_{A_0}$. By induction hypothesis there is a σ-embedding g_0 on $\mathbb{Q}^*$ which extends f_0. Let $X = \{x_1, \ldots, x_m\}$ with $x_i \sqsubseteq x_{i+1}$ for all $1 \leq i < m$. As A is closed under maximal common ancestors, there are rational numbers $q_1 < \cdots < q_m$ and words $u_1, \ldots, u_m$ such that $x_i = y q_i u_i$. For any two indices $1 \leq i < j \leq m$, we have $x_i \sqcap x_j = y$ and so $f(x_i) \sqcap f(x_j) = f(y)$, as f is compatible with $\sqcap$. Since $f(y)$ is the maximal common prefix of any two values x_j and x_j, there are rational numbers $q_1' < \cdots < q_m'$ and words $u_1', \ldots, u_m'$ with $f(x_i) = f(y) q_i' u_i'$. Next, choose a bijective, strictly monotonic function $\ell \colon \mathbb{Q} \to \mathbb{Q}$ with $\ell(q_i) = q_i'$ for all $1 \leq i \leq m$. For every $i \in \{1, \ldots, m\}$ let $f_i \colon \{\varepsilon, u_i\} \to \mathbb{Q}^*$ be given by $f_i(u_i) = u_i'$. Using the case $|A| \leq 2$ we obtain a σ-embedding $g_i \colon \mathbb{Q}^* \to \mathbb{Q}^*$ which extends f_i. We define g by

$$g(w) = \begin{cases} f(y)\, q_i'\, g_i(u) & \text{if } w = y q_i u \text{ for some } i \in \{1, \ldots, m\}, \\ f(y)\, \ell(q)\, u & \text{if } w = y q u \text{ and } q \notin \{q_1, \ldots, q_m\}, \\ g_0(w) & \text{if } w \not\prec z. \end{cases}$$

Using the notation of Lemma 9.6, we can represent g as follows:

$$g = g_0\big[y \leftarrow \widetilde{\ell}[q_i \leftarrow g_i]_{i=1,\ldots,m}\big].$$

By the choice of $g_0, \ldots, g_m$, g is an extension of f. Furthermore, g is a σ-embedding by Lemma 9.6. ∎

We are now ready to proof the strong upwards compatibility of $\to$ and $\to^{-1}$ with respect to $\leq_C$, where $\to^{-1}$ is the inverse relation i.e., $(p, \overline{u}) \to^{-1} (q, \overline{v})$ if and only if $(q, \overline{v}) \to (p, \overline{u})$.

Proposition 9.8. $\to$ and $\to^{-1}$ are strongly upwards compatible with respect to $\leq_C$.

Proof. Given k-tuples $\overline{w}$, $\overline{v}$, $\overline{w}'$ and states q and p such that there is a transition $(q, \overline{w}) \to (p, \overline{v})$ and such that $\overline{w} \leq_C \overline{w}'$ we have to show that there is some $\overline{v}'$ such that $\overline{v} \leq_C \overline{v}'$ and $(q, \overline{w}') \to (p, \overline{v}')$.

Since $\overline{w} \leq_C \overline{w}'$, the isomorphism $h : \mathrm{MCAT}_C(\overline{w}) \to \mathrm{MCAT}_C(\overline{w}')$ extends (by Lemma 9.7) to a stretching $\{\preceq, \sqsubseteq, \sqcap\}$-embedding $\hat{h} : \mathbb{Q}^* \to \mathbb{Q}^*$. Setting $v_i' = \hat{h}(v_i)$ for each $v_i \in \overline{v}$ we obtain with $\overline{v}' = (v_1', \ldots, v_k')$ that $(p, \overline{v}) \leq_C (p, \overline{v}')$ and $(q, \overline{w}') \to (p, \overline{v}')$ as desired.

The argument for $\to^{-1}$ is completely analogous. ∎

We now consider a particular $(\preceq, \sqsubseteq, \sqcap, S)$-embedding: the insertion of an m-gap at some u which is not prefixed by a constant from C. This preserves the type and leads to a $\leq_C$ larger tuple.

Definition 9.9. Let u be a word and $m \in \mathbb{N}$. We define the *insertion of an m-gap at u* to be $\iota_u^m : \mathbb{Q}^* \to \mathbb{Q}^*$ given by $\iota_u^m(w) = u0^m v$ if $w = uv$ and $\iota_u^m(w) = w$ if $u \not\preceq w$.

Clearly, ι_u^m is also a stretching function. Hence, it preserves $\leq_C$ on the configurations. Iterated use of this fact and Proposition 9.8 proves the following lemma.

Lemma 9.10. Given two configurations $(q, \overline{w})$, $(q, \overline{v})$ such that $\mathrm{typ}_C(\overline{w}) = \mathrm{typ}_C(\overline{v})$ then there is a configuration $(q, \overline{u})$ such that $(q, \overline{w}) \leq_C (q, \overline{u})$ and $(q, \overline{v}) \leq_C (q, \overline{u})$.

Proof. Let $d \in \mathbb{N}$ be maximal such that there are $x_1, x_2 \in \mathrm{MCAT}_C(\overline{w})$ with $x_1 \preceq x_2$ and $|x_2| - |x_1| = d$. Inductively, from the $\preceq$-maximal elements to ε we insert a d-gap at each $y \in \mathrm{MCAT}_C(\overline{v})$ if y is not prefixed by a constant from C. All these iterated insertions result finally in a tuple $\overline{u}$ such that $(q, \overline{v}) \leq_C (q, \overline{u})$ and for all $z_1, z_2 \in \mathrm{MCAT}_C(\overline{u})$ such that $z_1 \preceq z_2$ and z_2 is not prefix of any constant from C, then $|z_2| - |z_1| \geq d$. Thus, by definition of d also $(q, \overline{w}) \leq_C (q, \overline{u})$ holds as desired. ∎

We are finally ready to characterise the non-emptiness of $\mathcal{T}_{\mathbb{Q}}^C$-constraint automata by the existence of particular loops.

Definition 9.11. A *loop* is a finite run $r = (c_i)_{i \leq n}$ with $c_0 = (q, \overline{w})$, $c_n = (q, \overline{v})$ and $\mathrm{typ}_C(\overline{w}) = \mathrm{typ}_C(\overline{v})$. We say that a loop $r = (c_i)_{i \leq n}$ is *stretching* if $c_0 \leq_C c_n$.

Lemma 9.12. Let A be a constraint automaton. A has an accepting run if and only if there are finite runs r_1, r_2 where r_1 starts in an initial configuration and ends in some configuration c whose state is a final state, and where r_2 is a stretching loop starting in c.

Proof. ($\Rightarrow$). Let $r = (c_i)_{i\in\mathbb{N}}$ be an accepting run. Since r contains infinitely many configurations with a final state and $\leq_C$ is a wqo, we can find numbers $n_1 < n_2$ such that $c_{n_1} \leq_C c_{n_2}$ whence $(c_n)_{n\leq n_1}$, $(c_n)_{n_1\leq n\leq n_2}$ are the desired runs.
($\Leftarrow$). Assume r_1 is a run from some initial configuration to c_1 whose state is a final state $f \in F$ and r_2 is a stretching loop starting in c_1 and ending in c_2. Since $c_1 \leq_C c_2$, iterated use of strong upwards compatibility (Proposition 9.8) yields runs r_i from c_{i-1} to c_i such that $c_{i-1} \leq_C c_i$ for all $i \geq 3$. Clearly, the composition of $r_1, r_2, r_3, r_4, \ldots$ is an accepting run. ∎

9.2 Stretching Loops and Types of Runs

The last subsection provided a characterisation of loops using concrete data values. In order to obtain a decision procedure we abstract from these concrete values in this subsection. We give a characterisation of loops that lead to an accepting run, which only depends on the relations between the data values.

Definition 9.13. Let $r = (c_i)_{0\leq i\leq n}$ be a finite run, with $c_0 = (q, \overline{w})$ and $c_n = (p, \overline{v})$. Setting $\pi = \mathrm{typ}_C(\overline{w}, \overline{v})$, we say r has *type* $\mathrm{typ}(r) = (q, \pi, p)$.

Definition 9.14. Let $\overline{w}, \overline{v}$ be k-tuples of words such that $\mathrm{typ}_C(\overline{w}) = \mathrm{typ}_C(\overline{v})$ and let h be the induced isomorphism from $\mathrm{MCAT}_C(\overline{w})$ to $\mathrm{MCAT}_C(\overline{v})$. $(\overline{w}, \overline{v})$ is called *contracting* if one of the following holds.

1. There is some $d \in \mathrm{MCAT}_C(\overline{w})$ such that $h(d) \prec d$.
2. There are $d, e \in \mathrm{MCAT}_C(\overline{w})$ such that $d \prec e$, $h(e) = e$ and $d \prec h(d)$.

We call a loop r from $(q, \overline{w})$ to $(q, \overline{v})$ contracting if $(\overline{w}, \overline{v})$ is contracting. Otherwise, we call it (and its type) noncontracting.

Remark 9.15. As contracting only depends on the relations $=, \leq$, and $\sqsubseteq$ and not on the actual values of the positions in $\mathrm{MCAT}_C(\overline{w})$ and $\mathrm{MCAT}_C(\overline{v})$, it only depends on $\mathrm{typ}_C(\overline{w}, \overline{v})$ whether $(\overline{w}, \overline{v})$ is contracting.

Let us explain the term "contracting". Fix a loop from $(q, \overline{w})$ to $(q, \overline{v})$. The isomorphism $h : \mathrm{MCAT}_C(\overline{w}) \to \mathrm{MCAT}_C(\overline{v})$ relates for every pair $x \preceq y$ with $x, y \in \mathrm{MCAT}_C(\overline{w})$ the interval (x, y) with the interval $(h(x), h(y))$. By definition, for every contracting loop there is an interval (x, y) such that $|y| - |x| > |h(y)| - |h(x)|$.

The technical core of this section shows that if an automaton admits a noncontracting loop then it admits a stretching loop with the same initial and final state. This allows to rephrase the conditions from Lemma 9.12 in terms of types.

For runs $r = (c_i)_{i\in I}$ and $r' = (d_i)_{i\in I}$ we write $r \leq_C r'$ if $c_i \leq_C d_i$ for all $i \in I$.

Recall from Proposition 8.8, that given a $(\preceq, \sqsubseteq, S)$-embedding f the sequence $r = \big((q_i, \overline{w}_i)\big)_{i\geq 1}$ is a run in A if and only if $f(r) = \big((q_i, f(\overline{w}_i))\big)_{i\geq 1}$ is a run in A. In particular, this holds if $f = \iota_u^m$, i.e., the insertion of an m-gap at position u.

Let $w, v \in \mathbb{Q}^*$. We say that w and v are *comparable* if $w \preceq v$ or $v \preceq w$ holds. Otherwise, we call u and v *incomparable*. In this situation, we distinguish two cases: we say w is *incomparable left* of v if $w \sqsubseteq v$ and $w \not\preceq v$. In the same situation we call v *incomparable right* of w.

Proposition 9.16. Let r be a noncontracting loop. Then, there is a stretching loop r' such that $r \leq_C r'$.

Proof. Let r from $(q, \overline{w})$ to $(q, \overline{v})$ be a noncontracting loop and $h : \mathrm{MCAT}_C(\overline{w}) \to \mathrm{MCAT}_C(\overline{v})$ the induced isomorphism. We iteratively define a sequence $r = r_0 \leq_C r_1 \leq_C \cdots \leq_C r_n$ of runs until r_n is stretching.

We call a pair $(u_1, u_2) \in \mathrm{MCAT}_C(\overline{w})^2$ *problematic* (with respect to r) if $u_1 \preceq u_2$ and $|u_2| - |u_1| > |h(u_2)| - |h(u_1)|$. Recall that in this case u_2 and $h(u_2)$, respectively, are not prefixes of any constant c from C because h fixes all such elements and C. Let P_r be the set of all problematic pairs. We split the set of all problematic pairs into three parts, which we handle separately (cf. Figure 9.2 for an example). Let

$$\begin{aligned} L_r &= \{(u_1, u_2) \in P_r \mid u_2 \text{ incomparable left of } h(u_2)\}, \\ R_r &= \{(u_1, u_2) \in P_r \mid u_2 \text{ incomparable right of } h(u_2)\}, \text{ and} \\ D_r &= \{(u_1, u_2) \in P_r \mid u_2 \text{ comparable to } h(u_2)\}. \end{aligned}$$

L-Step: If L_r is nonempty, choose the $\sqsubseteq$-minimal u_2 such that there is u_1 with $(u_1, u_2) \in L_r$. Now fix u_1 such that $(u_1, u_2) \in L_r$ and $d := (|u_2|-|u_1|)-(|h(u_2)|-|h(u_1)|)$ is maximal. Let $\iota = \iota^d_{h(u_2)}$ be the insertion of a d gap at $h(u_2)$ and $r' = \iota(r)$. Denote by $\iota(\overline{w})$ ($\iota(\overline{v})$) the data values of the first (last, respectively) configuration of r'. Let $h' : \mathrm{MCAT}_C(\iota(\overline{w})) \to \mathrm{MCAT}_C(\iota(\overline{v}))$ be the corresponding isomorphism. By definition the set $L_{r'} = \{(x_1, x_2) \in P_{r'} \mid x_2 \text{ incomparable left of } h'(x_2)\}$ does not contain a pair $(u, \iota(u_2))$ for any $u \in \mathrm{MCAT}_C(\iota(\overline{w}))$. Nevertheless, r' may admit problematic pairs that are not problematic with respect to r. This can happen if there are $x_1, x_2 \in \mathrm{MCAT}_C(\overline{w})$ such that $x_1 \prec h(u_2) \preceq x_2$ holds, but $h(x_1) \prec h(u_2) \preceq h(x_2)$ does not. Then, the distance between $\iota(x_1)$ and $\iota(x_2)$ is greater than the distance between x_1 and x_2 (by d). On the other hand, either both or none of $h'(\iota(x_1))$ and $h'(\iota(x_2))$ are shifted by the insertion of the gap whence their distance is equal to the distance of $h(x_1)$ and $h(x_2)$.

In this case, possibly $(\iota(x_1), \iota(x_2))$ is problematic w.r.t. r' while (x_1, x_2) is not problematic w.r.t r. Since u_2 is incomparable left of $h(u_2)$ and $h(u_2) \prec x_2$, we have

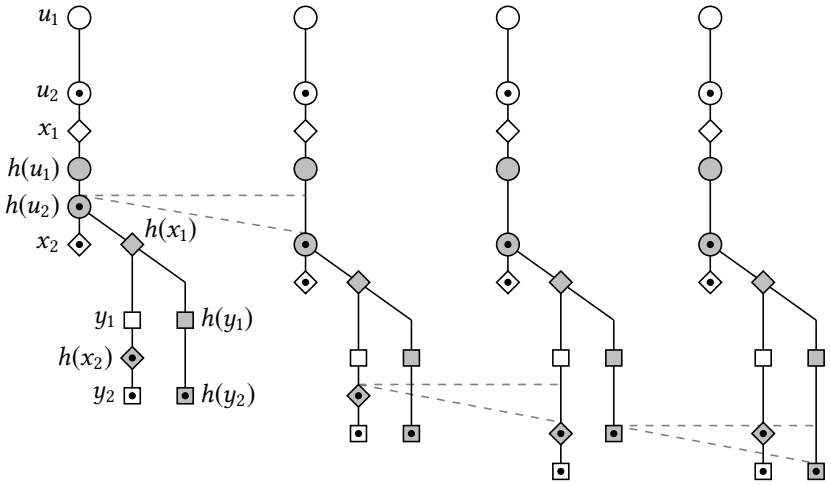


Figure 9.2: Example for Proposition 9.16: In the first tree (u_1, u_2) is problematic, insertion of a gap (D-Step) at $h(u_2)$ makes (the pair corresponding to) (x_1, x_2) problematic; insertion of a gap (L-Step) at $h(x_2)$ makes (y_1, y_2) problematic; insertion of a gap (L-Step) at $h(y_2)$ makes the tree stretching.

that u_2 is incomparable left of x_2 and x_2 is incomparable left of $h(x_2)$. Whence the same holds for $\iota(x_2)$, $h'(\iota(x_2)) = \iota(h(x_2))$ and $\iota(u_2)$. Thus, if $(\iota(x_1), \iota(x_2))$ is problematic, then $(\iota(x_1), \iota(x_2)) \in L_{r'}$ and $\iota(u_2)$ is strictly incomparable left of $\iota(x_2)$.

Thus, iteration of this step only creates problematic pairs that are more and more to the right with respect to $\mathrm{typ}_C(\overline{w}_n) = \mathrm{typ}_C(\iota(\overline{w}))$. Since $\mathrm{typ}_C(\overline{w}_n)$ is finite, we eventually do not introduce new problematic pairs and obtain a run r_i such that $L_{r_i} = \emptyset$ and $r \leq_C r_i$ because r_i results from insertion of several gaps in r.

R-Step: If $R_r \neq \emptyset$, proceed as in (L-Step), but exchange "left" and "right".

D-Step: If $L_r = R_r = \emptyset$ and r is not stretching, then $D_r \neq \emptyset$. Choose $u_2 \sqsubseteq$-minimal in $\mathrm{MCAT}(\overline{w})$ such that there is some u_1 with $(u_1, u_2) \in D_r$ and choose $u_1 \prec u_2$ in $\mathrm{MCAT}_C(\overline{w})$ such that $d := (|u_2| - |u_1|) - (|h(u_1)| - |h(u_2)|)$ is maximal. Since r is not contracting we have $u_2 \preceq h(u_2)$ and $u_1 \preceq h(u_1)$. Assume $u_2 = h(u_2)$, then $u_1 \prec h(u_1)$ as $(u_1, u_2) \in D$. This contradicts that r is not contracting. Thus, $u_2 \prec h(u_2)$. Again, let $\iota = \iota^d_{h(u_2)}$ and $r' = \iota(r)$.

Define $\iota(\overline{w})$, $\iota(\overline{v})$ and h' as in the L-step. Again there may be a pair (x_1, x_2) which is not problematic with respect to r while $(\iota(x_1), \iota(x_2))$ is problematic with respect to r'. If $R_{r'}$ or $L_{r'}$ are nonempty, we can deal with those problematic intervals using R- or L-steps. This finally leads to a run r_j with $R_{r_j} = L_{r_j} = \emptyset$. Moreover, for every pair (x_1, x_2) such that this pair is not problematic with respect to r but $(\iota(x_1), \iota(x_2))$ is problematic with respect to r', we conclude that x_2 is strictly below u_2 whence

$\iota(x_2)$ is strictly below $\iota(u_2)$ w.r.t. $\preceq$. Thus, the endpoints of problematic pairs move downwards (in $\mathrm{typ}_C(\overline{w}, \overline{v}) = \mathrm{typ}_C(\overline{w}', \overline{v}')$) and eventually all problematic pairs are removed. Once r_j is a loop without problematic pairs, it is stretching. ∎

Corollary 9.17. The set of words accepted by an automaton A is nonempty if and only if there are runs r_1 and r_2 such that r_2 is a noncontracting loop starting in configuration $(f, \overline{w})$ where f is a final state and r_1 is a run from an initial configuration to some configuration $(f, \overline{v})$ such that $\mathrm{typ}_C(\overline{w}) = \mathrm{typ}_C(\overline{v})$.

Proof. Due to Lemma 9.12 and the fact that every stretching loop is also non-contracting, only ($\Leftarrow$) requires a proof. Assume that there are runs r_1, r_2 as stated above. By Lemma 9.10, there is a configuration c_0 with $(f, \overline{v}) \leq_C c_0$ and $(f, \overline{w}) \leq_C c_0$. Using Lemma 9.7, we obtain a stretching σ-embedding $g\colon \mathbb{Q}^* \to \mathbb{Q}^*$ which maps $(f, \overline{w})$ to c_0. Applying g to every configuration in r_2 results in a new run $r_2' \geq_C r_2$. As g is an σ-embedding, r_2' is also non-contracting. Whence by Proposition 9.16 there is a stretching loop r_2'' with $r_2' \leq_C r_2''$. This loop starts in some configuration c_1 such that $(f, \overline{v}) \leq_C c_1$. Applying Proposition 9.8 to r_1 and c_2 we obtain a run r_1' from an initial configuration to c_2. Thus, r_1' and r_2'' match the conditions of Lemma 9.12 which completes the proof. ∎

9.3 Computation of Types

In order to turn this characterisation of emptiness in terms of types into an effective algorithm for the emptiness problem the last missing step is to compute whether a given type is realised by some run of a given automaton. Let us first define the set of all types and the associated product operation.

Recall that B_n^C contains all propositional logic formulas where the atomic formulas are given by $v \sim v'$ with $v, v \in \{x_1, \ldots, x_n, y_1, \ldots, y_n\} \cup C$ and $\sim \in \{=, \preceq, \sqsubseteq\}$. We say an isomorphism type $\pi = \mathrm{typ}_C(\overline{w}, \overline{v})$ satisfies a formula $\beta \in B_n^C$, written $\pi \models \beta$, if $(\mathcal{T}_{\mathbb{Q}}^C, \overline{w}, \overline{v}) \models \beta$. Note that this definition is well-defined, i.e., if $\mathrm{typ}_C(\overline{w}, \overline{v}) = \mathrm{typ}_C(\overline{w}', \overline{v}')$ then $(\mathcal{T}_{\mathbb{Q}}^C, \overline{w}, \overline{v}) \models \beta$ if and only if $(\mathcal{T}_{\mathbb{Q}}^C, \overline{w}', \overline{v}') \models \beta$ as $\mathrm{MCAT}_C(\overline{w}, \overline{v})$ and $\mathrm{MCAT}_C(\overline{w}', \overline{v}')$ are isomorphic.

Definition 9.18. We make the following definitions:

1. Let $\mathrm{RunTypes}_n^C$ denote the *set of all types* (q, π, p) where $q, p \in Q$ and $\pi = \mathrm{typ}_C(\overline{w}, \overline{v})$ for some n-tuples of words $\overline{w}$ and $\overline{v}$.

2. We equip the power set $2^{\mathrm{RunTypes}_n^C}$ with a *product* $\cdot$ as follows: let $S, T \subseteq \mathrm{RunTypes}_n^C$, then $S \cdot T$ contains all types (p, π, q) such that there are words $\overline{u}, \overline{v}, \overline{w} \in (\mathbb{Q}^*)^n$ and a state $r \in Q$ with $(p, \mathrm{typ}_C(\overline{u}, \overline{v}), r) \in S$, $(r, \mathrm{typ}_C(\overline{v}, \overline{w}), q) \in T$, and $\pi = \mathrm{typ}_C(\overline{u}, \overline{w})$.

3. The *set of types of one-step runs* $T_1 \subseteq \mathrm{RunTypes}_n^C$ is given by $t = (q, \pi, p) \in T_1$ if there is a transition (q, β, p) of A such that π satisfies β.

Using the just introduced product operation, we define the iteration of an element as usual: $T^1 := T$ and $T^{n+1} := T^n \cdot T$. Furthermore, the set T^+ given by $T^+ := \bigcup_{n\geq 1} T^n$ contains all types that appear in some power of T. The product operation resembles the composition of types. As a consequence one can connect the runs of A and T_1^+.

Next, we show that for every run r its type is contained in T_1^+, i.e., $\mathrm{typ}(r) \in T_1^+$. We also show the converse direction, that every type $t \in T_1^+$ admits a run r with $\mathrm{typ}(r) = t$. Thus the elements of T_1^* are exactly the types of the runs of the automaton. We will later use this correspondence to check if an arbitrary type can be realised in the automaton, i.e., is the type of a run.

Lemma 9.19. For every run $r = (c_i)_{1\leq i\leq k}$ with $k \geq 1$, we have $\mathrm{typ}(r) \in T_1^{k-1}$.

Proof. For $k = 2$ the claim follows by definition of $T_1^{2-1} = T_1$. We proceed by induction. Write $c_i = (q_i, w_1^i, \ldots, w_\ell^i)$. Let $r' = (c_i)_{1\leq i\leq k-1}$ and $r_{k-1} = (c_i)_{k-1\leq i\leq k}$. By induction hypothesis $\mathrm{typ}(r') = (q_1, \pi, q_{k-1}) \in T_1^{k-2}$ with

$$\pi = \mathrm{typ}_C(w_1^1, w_2^1, \ldots, w_\ell^1, w_1^{k-1}, \ldots, w_\ell^{k-1}),$$

and $\mathrm{typ}(r_{k-1}) = (q_{k-1}, \pi_{k-1}, q_k) \in T_1$ with

$$\pi_{k-1} = \mathrm{typ}_C(w_1^{k-1}, \ldots, w_\ell^{k-1}, w_1^k, \ldots, w_\ell^k).$$

Thus, the tuples $w_1^1, \ldots, w_\ell^1, w_1^{k-1}, \ldots, w_\ell^{k-1}, w_1^k, \ldots, w_\ell^k$ witness that

$$(q_1, \pi', q_k) := \mathrm{typ}(r) \in \mathrm{typ}(r') \cdot \mathrm{typ}(r_{k-1}) \subseteq T_1^{k-2} \cdot T_1 = T_1^{k-1},$$

which completes the proof. ∎

Lemma 9.20. Let $k \geq 1$ and $t \in T_1^k$. There is a run $r = (c_i)_{i=1,\ldots,k+1}$ with $\mathrm{typ}(r) = t$.

Proof. We use induction over k. For $k = 1$, we have $t \in T_1$ and so $t = (p, \pi, q)$ such that there is a $(p, \beta, q) \in T$ and $\pi \models \beta$. Choose $\overline{u}, \overline{v}$ with $\mathrm{MCAT}_C(\overline{u}, \overline{v}) = \pi$. Then, $(p, \overline{u}) \to (q, \overline{v})$ is the desired run of length 2.

Assume $k > 1$. Let $t \in \{t_0\} \cdot \{t_1\}$ with $t_0 \in T_1^{k-1}$ and $t_1 \in T_1$. Let $t = (p, \pi, q)$, $t_0 = (p, \pi_0, r)$, and $t_1 = (r, \pi_1, q)$. By definition of the type product, there are tuples of words $\overline{x}$, $\overline{y}$, and $\overline{z}$ with $\pi_0 = \mathrm{typ}_C(\overline{x}, \overline{y})$, $\pi_1 = \mathrm{typ}_C(\overline{y}, \overline{z})$, and $\pi = \mathrm{typ}_C(\overline{x}, \overline{z})$. By induction hypothesis, there is a run $r_0 = (q_i, \overline{u}_i)_{i=1}^k$ with $\mathrm{typ}(r_0) = t_0$, i.e., $p_1 = p$, $p_k = r$, and $\mathrm{typ}_C(\overline{u}_1, \overline{u}_k) = \mathrm{typ}_C(\overline{x}, \overline{y})$.

Let $h\colon \mathrm{MCAT}_C(\overline{x},\overline{y}) \to \mathrm{MCAT}_C(\overline{u}_1,\overline{u}_k)$ be an isomorphism. We modify the image of h to obtain a stretching isomorphism. Let N be the maximal distance between to adjacent nodes in $\mathrm{MCAT}_C(\overline{x},\overline{y})$ and define the function $f\colon \mathbb{Q}^* \to \mathbb{Q}^*$ by

$$f(x) = c\ 0^N x_1' 0^N x_2' 0^N \cdots 0^N x_\ell' 0^N,$$

where $x = cx'$ with $c \in C$ maximal with $c \preceq x$ and $x' = x_1' \cdots x_\ell'$. Clearly, f is a $(\preceq, \sqsubseteq, \sqcap, S)$-embedding and the composition $f \circ h\colon \mathrm{MCAT}_C(\overline{x},\overline{y}) \to \mathbb{Q}^*$ is stretching. Moreover, we have $f(\mathrm{MCAT}_c(\overline{u})) = \mathrm{MCAT}_C(f(\overline{u}))$ for all tuples of words $\overline{u}$. By Lemma 9.7, there is a σ-embedding $h'\colon \mathbb{Q}^* \to \mathbb{Q}^*$ which extends $f \circ h$. Let h_1 the restriction of h' to $\mathrm{MCAT}_C(\overline{x},\overline{y},\overline{z})$ and define $\overline{w} = h_1(\overline{z})$. Thus, h_1 is a isomorphism $\mathrm{MCAT}_C(\overline{x},\overline{y},\overline{z}) \to \mathrm{MCAT}_C(f(\overline{u_1}), f(\overline{u_k}), \overline{w})$. Therefore, $\mathrm{typ}_C(\overline{x},\overline{y},\overline{z}) = \mathrm{typ}_C(f(\overline{u_1}), f(\overline{u_k}), \overline{w})$. Let $f(r) = (p_i, f(\overline{u}_i))_{i=1}^k$. Then $f(r)$ is also a run and $\mathrm{typ}(r) = \mathrm{typ}(f(r))$ holds as f is a σ-embedding. Furthermore, as $\mathrm{typ}_C(\overline{y},\overline{z}) = \mathrm{typ}_C(f(\overline{u}_k), \overline{w})$ and $t_1 \in T_1$, we conclude that $r' = (p_1, f(\overline{u}_1)) \cdots (p_k, f(\overline{u}_k))(q, \overline{w})$ is a run. As $\pi = \mathrm{typ}_C(\overline{x},\overline{z}) = \mathrm{typ}_C(f(\overline{u}_1), \overline{w})$, we obtain $\mathrm{typ}(r) = t$. ∎

From the last to lemmas we immediately conclude the following result.

Corollary 9.21. There is a finite run of A of type t if and only if $t \in T_1^+$.

9.4 Representation of Tree Types

Before we state our complexity result, we investigate how to efficiently store tree types in memory. Let $\pi = \mathrm{typ}_C(\overline{u})$ for some tuple of words $\overline{u}$. The naïve approach would just store every component of $\overline{u}$ as a list of pairs of integers. Unfortunately, the size of such a representation requires space which is not logarithmic in the size of the constants. There are two reasons for this:

1. A suffix u_i of a constant c includes the whole constant c in its naïve representation,

2. If c, cq_1, cq_3 are constants and $u = cq_2$ for $q_1, q_2, q_3 \in \mathbb{Q}$, the size of the integers representing q_2 might be linear in the size of q_1, q_3.

We fix the first issue by writing $u_i = c_i u_i'$ where $c_i \in C$ is maximal with $c_i \preceq u_i$ and only storing the *index of* c_i and u_i'. To overcome the second issue, we do not store the exact values of u_i', but any values v_i' with $\mathrm{MCAT}_C(u_1', \ldots, u_n') = \mathrm{MCAT}_C(v_1', \ldots, v_n')$. The values v_i' can be chosen to be always in $\{1, \ldots, n\}$. This transformation does not preserve the left-right-order of the v_i's with the constants. Therefore, we store for every i the maximal constant ℓ_i which is left of u_i and on the same level. Recall,

that the set of constants is closed under prefixes. This process results in a small representation of $\mathrm{typ}_C(\overline{u})$. This will be formalised and proven in the next lemma.

Note that any finite set $A \subseteq \mathbb{Q}^*$, which is closed under maximal common prefixes, can be $\{\preceq, \sqsubseteq, \sqcap\}$-embedded in a tree with branching degree at most $|A|$ and height at most $|A|$. Thus, the $\{\preceq, \sqsubseteq, \sqcap\}$-isomorphism class of $(A, \preceq, \sqsubseteq, \sqcap)$ can be represented in space at most $|A|^2 \log(|A|)$. Moreover, any set A' can be closed under maximal common prefixes by adding of at most $|A'|$ elements to A'.

Lemma 9.22. Let $n \geq 1$ and $\mathrm{TreeTypes}_n^C = \{\mathrm{typ}_C(\overline{u}) \mid \overline{u} \in (\mathbb{Q}^*)^n\}$. Moreover, let the set of representation R_n^C be given by

$$R_n^C = \left\{ ((c_i, \ell_i, v_i))_{i=1}^n \in \left(C^2 \times \{1, \ldots, n\}^{\leq n}\right)^n \;\middle|\; c_i \preceq \ell_i,\ |\ell_i| - |c_i| \leq 1 \text{ for all } i \right\},$$

where $\{1, \ldots, n\}^{\leq n} = \bigcup_{i=0}^n \{1, \ldots, n\}^i$. Then, there is a surjective function $h \colon R_n^C \to \mathrm{TreeTypes}_n^C$, such that for every $r \in R$ the relations $\preceq$ and $\sqsubseteq$ in $h(r)$ can be computed from r in logarithmic space.

Especially, it is possible to represent an element of $\mathrm{RunTypes}_n^C$ in space linear in $\log(|Q|) \cdot n(\log(|C|) + \log(m)) \cdot n \log(n)$ where $m = \max\{|c| \mid c \in C\}$.

We will use the rest of this section for the proof of this lemma.

Let $r = ((c_i, \ell_i, v_i))_{i=1}^n \in R_n^C$. For every $i = 1, \ldots, n$ choose words u_i as follows: if v_i is $\preceq$-minimal in $\{v_j \mid c_j = c_i,\ \ell_j = \ell_i\}$ assume $C \cap c_i\mathbb{Q} = \{c_i q_1, \ldots, c_i q_k\}$, if $\ell_i = c_i$ set $u_i = c_i(q_1 - 1)$ (or $u_i = c_i 1$ if $k = 0$). Otherwise, $\ell_i = c_i q_j$ and we define $u_i = c_i q$ with $q = 1/2\,(q_j + q_{j+1})$ (or just $q = q_j + 1$ if $j = k$). If v_i is not $\preceq$-minimal in $\{v_j \mid c_j = c_i,\ \ell_j = \ell_i\}$, let $v_i = v_j v'$, where v_j is minimal, and set $u_i = u_j v'$, where u_j is the element construction in the first case. We define h by $h(r) = \mathrm{typ}_C(u_1, \ldots, u_n)$.

The chosen elements $u_1, \ldots, u_n$ satisfy

$$\begin{gathered} c_i = \max_{\preceq}\{c \in C \mid c \preceq u_i\}, \qquad \ell_i = \max_{\sqsubseteq}\{\ell \in C \mid c_i \preceq \ell \sqsubseteq u_i,\ |\ell| - |c| \leq 1\}, \\ \mathrm{MCAT}(u_j' \mid c_j = c_i,\ \ell_j = \ell_i) \cong \mathrm{MCAT}(v_j \mid c_j = c_i, \ell_j = \ell_i), \end{gathered} \tag{9.1}$$

where $u_i = c_i u_i'$ for $i = 1, \ldots, n$. This can be seen directly from the definition of u_i's, but also makes use of the fact that C is closed under prefixes.

Before we show surjectivity, we argue that any representation $r = ((c_i, \ell_i, v_i))_{i=1}^n$ satisfying (9.1) carries enough information to reconstruct the relations $\preceq$ and $\sqsubseteq$ on $\mathrm{typ}_C(u_1, \ldots, u_n)$. We begin with $\preceq$:

Sublemma 9.23. $u_i \preceq u_j$ if and only if one of the following conditions holds:

1. $c_i \prec c_j$ and $v_i = \varepsilon$
2. $c_i = c_j$, $\ell_i = \ell_j$ and $v_i \preceq v_j$

Proof. We distinguish three cases:

$c_i = c_j$ Assume $u_i \preceq u_j$. Assume there is an $\ell \in C$ with $c_i \preceq \ell$, $\ell \sqsubseteq u_j$ but $u_i \sqsubset \ell$. Thus, $u_i \prec \ell \preceq u_j$, which contradicts the maximality of c_i. Therefore, $\ell \sqsubseteq u_i$ iff $\ell \sqsubseteq u_j$ and so $\ell_i = \ell_j$. Hence, $u_i' \preceq u_j'$ implies $v_i \preceq v_j$.

Conversely, assume $\ell_i = \ell_j$ and $v_i \preceq v_j$. This implies $u_i' \preceq u_j'$ and thus $u_i \preceq u_j$.

$c_i \prec c_j$ Assume $u_i \preceq u_j$. By maximality of c_i, we have $u_i' = \varepsilon$ in this case, and thus $v_i = \varepsilon$ (Recall that the MCAT always includes ε.)

Conversely, $v_i = \varepsilon$ implies $u_i' = \varepsilon$. Thus, $u_i \preceq u_j$.

$c_j \prec c_i$ By maximality of c_j, u_i cannot be a prefix of u_j in this case. Moreover, neither condition 1 nor 2 are satisfied. ∎

Next, we establish the corresponding result for $\sqsubseteq$.

Sublemma 9.24. $u_i \sqsubseteq u_j$ if and only if one of the following conditions holds:[1]

1. $c_i \parallel c_j$ and $c_i \sqsubseteq c_j$
2. $c_i = c_j$ and ($\ell_i \sqsubset \ell_j$ or ($\ell_i = \ell_j$ and $v_i \sqsubseteq v_j$))
3. $c_i \prec c_j$ and (($\ell_i \sqsubseteq c_j$ and $\ell_i \parallel c_j$) or $\ell_i = c_i$)
4. $c_i \succ c_j$ and (($c_i \sqsubseteq \ell_j$ and $\ell_j \parallel c_i$) or $c_j \prec \ell_j \preceq c_i$)

Proof. We consider four cases:

$c_i \parallel c_j$ We have $u_i \sqsubseteq u_j \iff c_i \sqsubseteq c_j \iff$ condition 1. Thus the claim holds.

$c_i = c_j$ Assume $u_i \sqsubseteq u_j$ and $\ell_j \sqsubseteq \ell_i$. We have $\ell_i \sqsubseteq u_i \sqsubseteq u_j$ and so $\ell_i \sqsubseteq \ell_j$. This implies $\ell_i = \ell_j$. Thus, we obtain $u_i' \sqsubseteq u_j'$, as $u_j \sqsubseteq u_j$.

Conversely, assume condition 2. For $\ell_i \sqsubset \ell_j$, assume $u_j \sqsubset u_i$. Thus $\ell_j \sqsubseteq \ell_i$, a contradiction. Otherwise, $\ell_i = \ell_j$ and $v_i \sqsubseteq v_j$ and hence $u_i' \sqsubseteq u_j'$.

$c_i \prec c_j$ Assume $u_i \sqsubseteq u_j$. If $u_i' = \varepsilon$, we obtain $\ell_i = c_i$. We now consider the case $u_i' \neq \varepsilon$, i.e., $c_j \parallel u_i$. Assume $\ell_i \neq c_i$. Assume $c_j \sqsubset \ell_i$. Thus, $c_j \sqsubset u_i$ and $u_j \sqsubset u_i$. A contradiction. Hence, $\ell_i \sqsubseteq c_j$. Next, assume $c_i \prec \ell_i \prec c_j$. We have $\ell_i \sqsubset u_i$, as $\ell_i \parallel u_i$, and thus $c_j \sqsubset u_i$. A contradiction. Thus, $\ell_i \parallel c_j$.

Conversely, assume condition 3 holds. For $u_i' = \varepsilon$, we immediately obtain $u_i \sqsubseteq u_j$. Thus, assume $u_i' \neq \varepsilon$. Next, assume $\ell_i = c_i$. If $u_j \sqsubset u_i$, then $c_j \sqsubset u_i$ and

[1] $x \parallel y$ means $x \not\preceq y$ and $y \not\preceq x$.

thus there is a $c_i \prec c \prec c_j$ with $c \sqsubset u_i$, which contradicts the maximality of ℓ_i. Finally, let $\ell_i \parallel c_j$ and $\ell_i \sqsubseteq c_j$. If $u_j \sqsubseteq u_i$, then $c_j \sqsubseteq u_i$, thus there is a $c_i \prec c \prec c_j$ with $c \sqsubseteq u_i$. As $\ell_i \sqsubseteq c_j$ and $\ell_i \parallel c_j$, we have $\ell_i \sqsubset c$, which contradicts the maximality of ℓ_i. Therefore, $u_i \preceq u_j$.

$c_i \succ c_j$ Assume $u_i \sqsubseteq u_j$. First, assume $\ell_j \parallel c_i$. Assume $\ell_j \sqsubset c_i$. As $c_i \sqsubseteq u_j$ and $c_i \parallel u_j$, there is a $c_j \prec c \preceq c_i$ with $c \sqsubseteq u_j$. As $\ell_j \sqsubset c$, this contradicts maximality of ℓ_j. Hence, $c_i \sqsubseteq \ell_j$. Next, assume $\ell_j = c_j$. As before, there is a $c_j \prec c \preceq c_i$ with $c \sqsubseteq u_j$, this contradicts maximality of $\ell_j = c_j$.

Conversely, assume condition 4 holds. Assume $\ell_j \parallel c_i$ and $c_i \sqsubseteq \ell_j$. Thus, $u_i \sqsubseteq \ell_j \sqsubseteq u_j$. Next, assume $c_j \prec \ell_j \preceq c_i$. By maximality of c_j, we have $u_j \parallel \ell_j$ and thus $c_i \sqsubseteq u_j$, and so $u_i \sqsubseteq u_j$. Finally, assume $c_j \prec \ell_j \preceq c_i$. As $\ell_j \sqsubseteq u_j$ and $\ell_j \parallel u_j$, we obtain $c_i \sqsubseteq u_j$ and so $u_i \sqsubseteq u_j$. ∎

Thus, using the above two Sublemma, we obtain that if $r = ((c_i, \ell_i, v_i))_{i=1}^n \in R$ is a representation and $u_1, \ldots, u_n \in \mathbb{Q}^*$ such that (9.1) is satisfied, we have $h(r) = \mathrm{typ}_C(u_1, \ldots, u_n)$.

We show surjectivity. Let $t = \mathrm{typ}_C(\overline{u})$ with $\overline{u} = (u_1, \ldots, u_n) \in (\mathbb{Q}^*)^n$ we define words c_i and ℓ_i, an v_i just by (9.1). Note that the c_i and ℓ_i are uniquely determined by (9.1) and values v_i can always be chosen as any n tree nodes can be represented as subtree of $\{1, \ldots, n\}^{\leq n}$ preserving $\preceq$ and $\sqsubseteq$. Let r be the representation $r = (c_1, \ell_1, v_1) \cdots (c_n, \ell_n, v_n)$. We show $h(r) = t$. Assume $h(r) = t' = \mathrm{typ}_C(\overline{w})$ where $\overline{w} = (w_1, \ldots, w_n)$. As $\overline{u}$ and $\overline{w}$ both satisfy (9.1) using this representation r, we obtain by the above two Sublemma that $\mathrm{typ}_C(\overline{u}) = \mathrm{typ}_C(\overline{w})$.

Moreover, the conditions given in Sublemma 9.23 and Sublemma 9.24 can be checked in logarithmic space given a representation r. Therefore, the proof of Lemma 9.22 is completed. ∎

9.5 Emptiness of Constraint Automata

We are ready to state our decision procedure for the emptiness of $\mathcal{T}^C_{\mathbb{Q}}$-constraint automata in this section. As preparatory step, we argue that it can be checked in logarithmic space whether a type is contained in the product of two singleton sets of types.

Proposition 9.25. There is a nondeterministic algorithm that, given three run types t_0, t_1, t_2 represented as in Lemma 9.22, checks in space linear in $n^K(\log(|A|) + \log(|C|) + \log(m))$ whether $t_0 \in \{t_1\} \cdot \{t_2\}$ for some fixed $K \in \mathbb{N}$.

Proof. Let R^C_m be the sets from the last chapter. Let $t_i = (p_i, \pi_i, q_i)$ for $i = 0, 1, 2$ with $p_i, q_i \in Q$ and $\pi_i \in \mathrm{TreeTypes}^C_n$. We assume that the π_i are represented as elements from R^C_n. The algorithm guesses a element from $r = R^C_{3n}$. Assume $h(r) = \mathrm{typ}_C(\overline{x}, \overline{y}, \overline{z})$ and then checks whether $p_0 = p_1$, $q_0 = q_2$, $q_1 = p_2$, and $\mathrm{typ}_C(\overline{x}, \overline{z}) = \pi_0$, $\mathrm{typ}_C(\overline{x}, \overline{y}) = \pi_1$ and $\mathrm{typ}_C(\overline{y}, \overline{z}) = \pi_2$. These checks can be carried out in logarithmic space, as the relations on representation, i.e., elements from R^C_m, can be decided in logarithmic space. Correctness follows directly from the surjectivity of h: If this algorithm accepts an input, then $\overline{x}, \overline{y}, \overline{z}$ are witnesses for the product. Conversely, if $t_0 \in \{t_1\} \cdot \{t_2\}$, then there are words $\overline{x}, \overline{y}, \overline{z}$ as above. As h is surjective the algorithm can guess a representation r with $h(r) = \mathrm{typ}_C(\overline{x}, \overline{y}, \overline{z})$.∎

We now prove the main theorem of this chapter, which we already stated at the beginning of the chapter.

Theorem 9.1. Let C be a set of constants and A an n-dimensional $\mathcal{T}^C_{\mathbb{Q}}$-constraint automaton. Let furthermore $m = \max\{|c| \mid c \in C\}$. It is decidable in space linear in $n^K(\log(m) + \log(|C|) + \log(|A|))$, for some global constant K independent of C and A, whether $\mathrm{L}(A) \neq \emptyset$.

Proof (Proof of Theorem 9.1). By Corollary 9.17 and Lemma 9.21 it suffices that the algorithm guesses a type (i, π, f) and a non-contracting type (f, π', f) such that i is an initial state, f is a final state, and the order type of the last elements of π coincides with the order type of the first elements of π', and then verifies whether these types are realised by actual runs.

This test is carried out as follows: First, guess an initial type $t_1 \in T_1$. Afterwards iteratively guess types t_{n+1} and one-step types $s_{n+1} \in T_1$, and verifying that $t_{n+1} \in \{t_n\} \cdot \{s_{n+1}\}$. In every step check whether $t_n = (i, \pi, f)$ or $t_n = (f, \pi', f)$. Note that after the completion of a single step, the space occupied by t_n can be reused for the next step. As the number of run types is exponential in $n(\log(|C|)+\log(m)+n\log(n))$, a counter requiring space linear in the same term is used to guarantee termination.∎

Using this result, we conclude the desired complexity of the model checking for cLTL.

Corollary 9.26. The model checking problem for cLTL is PSPACE-complete.

Proof. PSPACE-hardness follows directly, from the corresponding result for LTL model checking. We show containment in PSPACE. The algorithm runs the decision procedure for emptiness of $\mathcal{T}^C_{\mathbb{Q}}$-constraint automata from Theorem 9.1 on the automaton arising from the cLTL formula and the input automaton as laid out in section 8.4. Though this automaton has size exponential in the input, we can apply

the same trick as in [VW94] to obtain a PSPACE decision procedure: instead of constructing and storing the automaton explicitly, whenever the algorithm needs to guess a state or a transition, the algorithm actually guesses some arbitrary string (of polynomial length) and then verifies that this string represents a state or a transition. This verification can run in polynomial space. Furthermore, a state of this automaton can also be remembered in polynomial space. ∎

Finally, the question arises what the exact complexity of the emptiness problem is. It turns out that the use of an arbitrary number of dimensions separates between NL and PSPACE.

Proposition 9.27. The following statements hold:

1. The emptiness problem for $\mathcal{T}_{\mathbb{Q}}$-constraint automata is PSPACE-complete.

2. For any fixed $n \geq 1$, the emptiness problem for n-dimensional $\mathcal{T}_{\mathbb{Q}}$-constraint automata is NL-complete.

Proof. We start with statement 2: for fixed n, containment in NL is the statement of Theorem 9.1. Hardness follows by reducing from graph reachability.

We show statement 1. The proof is inspired by the proof of PSPACE-hardness of timed graph reachability given in [CY92]. We reduce the LBA (linear bounded automaton) word acceptance problem to emptiness of constraint automata.

Given a LBA A and an input word w, we construct a set of constants C and an $|w|$-dimensional $\mathcal{T}_{\mathbb{Q}}^{C}$-constraint automaton B. Let $A = (Q, \Sigma, \Gamma, T, q_0, F, \square)$, where Σ is the input alphabet, Γ is the tape alphabet, $T \subseteq Q \times \Gamma \times Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}, \mathrm{H}\}$ is the transition relation, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\square \in \Gamma$ is a dedicated blank symbol. We may assume that every symbol in the tape alphabet Γ of A occurs in at least one transition of A, thus, the size of the encoding of A is at least $|\Gamma|$. Moreover, we assume that the only transitions possible in a state in F are self-loops. Choose $|\Gamma|$ many distinct elements of the domain, i.e., $C = \{c_\gamma \mid \gamma \in \Gamma\}$. This set can be computed in P, for example choose $C = \{1^i \mid i \leq |\Gamma|\}$.

The automaton B keeps track of the LBA's state and head position in its state space, and uses the values of the $|w|$ dimensions to remember the tape contents. More formally: $B = (Q', I', F', T')$ where $\iota \notin Q$ is a new symbol and

$$Q' = \{\iota\} \cup Q \times \{1, \ldots, |w|\}, \qquad I = \{\iota\}, \qquad F' = F \times \{1, \ldots, |w|\},$$
$$T' = \{(\iota, \alpha, (q_0, 1)\} \cup \{((q, n), \beta_{n,\gamma,\gamma'}, (q', n')) \mid \exists (q, \gamma, q', \gamma', d) \in T \colon n' = n + \epsilon(d)\},$$

where $\epsilon(d) = 1, -1, 0$ for $d = \mathrm{R}, \mathrm{L}, \mathrm{H}$, respectively, and

$$\alpha = \bigwedge_{i=1,\ldots,|w|} (y_i = c_{w_i}), \qquad \beta_{n,\gamma,\gamma'} = (x_n = c_\gamma) \wedge (y_n = c_{\gamma'}) \wedge \bigwedge_{\substack{i=1,\ldots,|w| \\ i \neq n}} (x_i = y_i).$$

Clearly, there is a one-to-one correspondence between the configurations of A and B (except for ι). Moreover, this correspondence is compatible with the respective transition relations. Thus, we have $(q, n, \overline{u}) \to_A (q', n', \overline{u}')$ if and only if $((q, n), \overline{v}) \to_B ((q', n'), \overline{v}')$, where $u_i = \gamma$ iff $v_i = c_\gamma$ for all $1 \leq i \leq n$ and $\overline{u} = (u_1, \ldots, u_n)$, $\overline{v} = (v_1, \ldots, v_n)$.

The additional state ι checks if $\overline{u}_2$ in a input data word $(\overline{u}_i)_{i \geq 1}$ encodes the word w and moves to the initial configuration of A. As A enters a loop around a final state when accepting a word, an accepting configuration of A translates to an infinite, accepting run in B and vice versa. ∎

Bibliography

[B07] M. Bojańczyk. 'Forest expressions'. In: *Computer Science Logic (CSL 2007)*. Vol. 4646. LNCS. Springer Berlin Heidelberg, 2007, pp. 146–160. DOI: 10.1007/978-3-540-74915-8_14.

[B60] J. R. Büchi. 'Weak second-order arithmetic and finite automata'. In: *Z. Math. Logik und Grundl. Math.* 6 (1960), pp. 66–92.

[B62] J. R. Büchi. 'On a decision method in restricted second order arithmetic'. In: *Logic, Methodology and Philosophy of Science*. Stanford Univ. Press, Stanford, Calif., 1962, pp. 1–11.

[BBG08] C. Baier, N. Bertrand and M. Grösser. 'On decision problems for probabilistic büchi automata'. In: *Foundations of Software Science and Computational Structures (FoSSaCS 2008)*. Vol. 4962. LNCS. Springer Berlin Heidelberg, 2008, pp. 287–301. DOI: 10.1007/978-3-540-78499-9_21.

[BD15] P. Babari and M. Droste. 'A nivat theorem for weighted picture automata and weighted mso logic'. In: *Language and Automata Theory and Applications (LATA 2015)*. Vol. 8977. LNCS. Springer International Publishing, 2015, pp. 703–715. DOI: 10.1007/978-3-319-15579-1_55.

[BG05] C. Baier and M. Grösser. 'Recognizing ω-regular languages with probabilistic automata'. In: *Logic in Computer Science (LICS 2005)*. 2005, pp. 137–146. DOI: 10.1109/LICS.2005.41.

[BG06] L. Bozzelli and R. Gascon. 'Branching-time temporal logic extended with qualitative presburger constraints'. In: *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2006)*. Vol. 4246. LNCS. Springer Berlin Heidelberg, 2006, pp. 197–211. DOI: 10.1007/11916277_14.

[BGMZ12] B. Bollig, P. Gastin, B. Monmege and M. Zeitoun. 'A probabilistic Kleene theorem'. In: *Automated Technology for Verification and Analysis (ATVA 2012)*. Vol. 7561. LNCS. Springer Berlin Heidelberg, 2012, pp. 400–415. DOI: 10.1007/978-3-642-33386-6_31.

[BK08] C. Baier and J.-P. Katoen. *Principles of Model Checking*. Representation and Mind Series. The MIT Press, 2008.

[BP14] L. Bozzelli and S. Pinchinat. 'Verification of gap-order constraint abstractions of counter systems'. In: *Theoretical Computer Science* 523 (2014), pp. 1–36. DOI: 10.1016/j.tcs.2013.12.002.

[BSSS06] M. Bojańczyk, M. Samuelides, T. Schwentick and L. Segoufin. 'Expressive power of pebble automata'. In: *Automata, Languages and Programming (ICALP 2006)*. Vol. 4051. LNCS. Springer Berlin Heidelberg, 2006, pp. 157–168. DOI: 10.1007/11786986_15.

[BT12] M. Bojanczyk and S. Torunczyk. 'Weak MSO+U over infinite trees'. In: *Symposium on Theoretical Aspects of Computer Science (STACS 2012)*. Vol. 14. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 648–660. DOI: 10.4230/LIPIcs.STACS.2012.648.

[Č94] K. Čerāns. 'Deciding properties of integral relational automata'. In: *Automata, Languages and Programming (ICALP 1994)*. Vol. 820. LNCS. Springer Berlin Heidelberg, 1994, pp. 35–46. DOI: 10.1007/3-540-58201-0_56.

[CDH09] K. Chatterjee, L. Doyen and T. A. Henzinger. 'Probabilistic weighted automata'. In: *CONCUR 2009 - Concurrency Theory*. Vol. 5710. LNCS. Springer Berlin Heidelberg, 2009, pp. 244–258. DOI: 10.1007/978-3-642-04081-8_17.

[CFKL15] C. Carapelle, S. Feng, A. Kartzow and M. Lohrey. 'Satisfiability of ECTL* with tree constraints'. In: *Computer Science – Theory and Applications (CSR 2015)*. Vol. 9139. LNCS. Springer International Publishing, 2015, pp. 94–108. DOI: 10.1007/978-3-319-20297-6_7.

[CH10] K. Chatterjee and T. A. Henzinger. 'Probabilistic automata on infinite words: decidability and undecidability results'. In: *Automated Technology for Verification and Analysis (ATVA 2010)*. Vol. 6252. LNCS. Springer Berlin Heidelberg, 2010, pp. 1–16. DOI: 10.1007/978-3-642-15643-4_1.

[CHS14] A. Carayol, A. Haddad and O. Serre. 'Randomization in automata on infinite trees'. In: *ACM Trans. Comput. Logic* 15.3 (2014), 24:1–24:33. DOI: 10.1145/2629336.

[CK12] C. C. Chang and H. J. Keisler. *Model Theory*. Dover Books on Mathematics. Dover Publications, 2012.

[CKL] C. Carapelle, A. Kartzow and M. Lohrey. 'Satisfiability of ECTL* with constraints'. under submission, draft available. URL: http://www.eti.uni-siegen.de/ti/veroeffentlichungen/15-ectl-constraints.pdf.

[CKL13] C. Carapelle, A. Kartzow and M. Lohrey. 'Satisfiability of CTL* with constraints'. In: *CONCUR 2013 – Concurrency Theory*. Vol. 8052. LNCS. Springer Berlin Heidelberg, 2013, pp. 455–469. DOI: 10.1007/978-3-642-40184-8_32.

[CLSV06] L. Cheung, N. Lynch, R. Segala and F. Vaandrager. 'Switched PIOA: parallel composition via distributed scheduling'. In: *Theoretical Computer Science* 365.1–2 (2006). Formal Methods for Components and Objects Formal Methods for Components and Objects, pp. 83–108. DOI: 10.1016/j.tcs.2006.07.033.

[CSV11] R. Chadha, A. Sistla and M. Viswanathan. 'Probabilistic büchi automata with non-extremal acceptance thresholds'. In: *Verification, Model Checking, and Abstract Interpretation (VMCAI 2011)*. Vol. 6538. LNCS. Springer Berlin Heidelberg, 2011, pp. 103–117. DOI: 10.1007/978-3-642-18275-4_9.

[CT12] K. Chatterjee and M. Tracol. 'Decidable problems for probabilistic automata on infinite words'. In: *Logic in Computer Science (LICS 2012)*. IEEE Computer Society, 2012, pp. 185–194. DOI: 10.1109/LICS.2012.29.

[CY92] C. Courcoubetis and M. Yannakakis. 'Minimum and maximum delay problems in realtime systems'. In: *Computer Aided Verification (CAV 1991)*. Vol. 575. LNCS. Springer Berlin Heidelberg, 1992, pp. 399–409. DOI: 10.1007/3-540-55179-4_37.

[D65] J. Doner. 'Decidability of the weak second-order theory of two successors'. In: *Notices Amer. Math. Soc.* 12 (1965), pp. 365–468.

[D70] J. Doner. 'Tree acceptors and some of their applications'. In: *Journal of Computer and System Sciences* 4.5 (1970), pp. 406–451. DOI: 10.1016/S0022-0000(70)80041-1.

[DD07] S. Demri and D. D'Souza. 'An automata-theoretic approach to constraint LTL'. In: *Information and Computation* 205.3 (2007), pp. 380–415. DOI: 10.1016/j.ic.2006.09.006.

[DD15] S. Demri and M. Deters. 'Temporal logics on strings with prefix relation'. In: *Journal of Logic and Computation* (2015). DOI: 10.1093/logcom/exv028.

[DG05] M. Droste and P. Gastin. 'Weighted automata and weighted logics'. In: *Automata, Languages and Programming (ICALP 2005)*. Vol. 3580. LNCS. Springer Berlin Heidelberg, 2005, pp. 513–525. DOI: 10.1007/11523468_42.

[DG08] S. Demri and R. Gascon. 'Verification of qualitative $\mathbb{Z}$ constraints'. In: *Theoretical Computer Science* 409.1 (2008), pp. 24–40. DOI: 10.1016/j.tcs.2008.07.023.

[DM10] M. Droste and I. Meinecke. 'Describing average- and longtime-behavior by weighted MSO logics'. In: *Mathematical Foundations of Computer Science (MFCS 2010)*. Vol. 6281. LNCS. Springer, 2010, pp. 537–548. DOI: 10.1007/978-3-642-15155-2_47.

[DM11] M. Droste and I. Meinecke. 'Weighted automata and regular expressions over valuation monoids'. In: *International Journal of Foundations of Computer Science* 22.8 (2011), pp. 1829–1844. DOI: 10.1142/S0129054111009069.

[DP14] M. Droste and V. Perevoshchikov. 'A nivat theorem for weighted timed automata and weighted relative distance logic'. In: *Automata, Languages, and Programming (ICALP 2014)*. Vol. 8573. LNCS. Springer Berlin Heidelberg, 2014, pp. 171–182. DOI: 10.1007/978-3-662-43951-7_15.

[DPV05] M. Droste, C. Pech and H. Vogler. 'A Kleene theorem for weighted tree automata'. In: *Theory of Computing Systems* 38.1 (2005), pp. 1–38. DOI: 10.1007/s00224-004-1096-z.

[DV06] M. Droste and H. Vogler. 'Weighted tree automata and weighted logics'. In: *Theor. Comput. Sci.* 366.3 (2006), pp. 228–247. DOI: 10.1016/j.tcs.2006.08.025.

[DV11] M. Droste and H. Vogler. 'Weighted logics for unranked tree automata'. In: *Theory of Computing Systems* 48.1 (2011), pp. 23–47. DOI: 10.1007/s00224-009-9224-4.

[E61] C. C. Elgot. 'Decision problems of finite automata design and related arithmetics'. In: *Transactions of the American Mathematical Society* 98.1 (1961), pp. 21–51. URL: http://www.jstor.org/stable/1993511.

[E71] C. A. Ellis. 'Probabilistic tree automata'. In: *Information and Control* 19.5 (1971), pp. 401–416. DOI: 10.1016/S0019-9958(71)90673-5.

[FGO12] N. Fijalkow, H. Gimbert and Y. Oualhadj. 'Deciding the value 1 problem for probabilistic leaktight automata'. In: *Logic in Computer Science (LICS 2012)*. IEEE Computer Society, 2012, pp. 295–304. DOI: 10.1109/LICS.2012.40.

[G09] R. Gascon. 'An automata-based approach for CTL* with constraints'. In: *Electronic Notes in Theoretical Computer Science* 239 (2009). Joint Proceedings of the 8th, 9th, and 10th International Workshops on Verification of Infinite-State Systems (INFINITY 2006, 2007, 2008), pp. 193–211. DOI: 10.1016/j.entcs.2009.05.040.

[GMMS14] T. Gogacz, H. Michalewski, M. Mio and M. Skrzypczak. 'Measure properties of game tree languages'. In: *Mathematical Foundations of Computer Science (MFCS 2014)*. Vol. 8634. LNCS. Springer Berlin Heidelberg, 2014, pp. 303–314. DOI: 10.1007/978-3-662-44522-8_26.

[GO10] H. Gimbert and Y. Oualhadj. 'Probabilistic automata on finite words: decidable and undecidable problems'. In: *Automata, Languages and Programming (ICALP 2010)*. Vol. 6199. LNCS. Springer Berlin Heidelberg, 2010, pp. 527–538. DOI: 10.1007/978-3-642-14162-1_44.

[K08] A. Klenke. *Probability Theory. A Comprehensive Course*. Universitext. Springer London, 2008. DOI: 10.1007/978-1-84800-048-3.

[K51] K. Knopp. *Theory and Application of Infinite Series*. Dover Books on Mathematics. Dover Publications, 1951.

[K56] S. C. Kleene. 'Representation of events in nerve nets and finite automata'. In: *Automata Studies*. Princeton University Press, 1956, pp. 3–42.

[KW15] A. Kartzow and T. Weidner. 'Model checking constraint LTL over trees'. In: *CoRR* abs/1504.06105 (2015). URL: http://arxiv.org/abs/1504.06105.

[L94] O. Louscou-Bozapalidou. 'Stochastic tree functions'. In: *International Journal of Computer Mathematics* 52.3-4 (1994), pp. 149–160. DOI: 10.1080/00207169408804300.

[M13] B. Monmege. 'Specification and Verification of Quantitative Properties: Expressions, Logics, and Automata'. PhD thesis. ENS Cachan, 2013.

[M63] D. E. Muller. 'Infinite sequences and finite machines'. In: *Switching Circuit Theory and Logical Design (SWCT 1963)*. IEEE Computer Society, 1963, pp. 3–16. DOI: 10.1109/SWCT.1963.8.

[M66] R. McNaughton. 'Testing and generating infinite sequences by a finite automaton'. In: *Information and Control* 9.5 (1966), pp. 521–530. DOI: 10.1016/S0019-9958(66)80013-X.

[MM70] M. Magidor and G. Moran. 'Probabilistic tree automata and context free languages'. In: *Israel Journal of Mathematics* 8.4 (1970), pp. 340–348. DOI: 10.1007/BF02798680.

[MMST02] L. Mora-López, R. Morales, M. Sidrach de Cardona and F. Triguero. 'Probabilistic finite automata and randomness in nature: a new approach in the modelling and prediction of climatic parameters'. In: *Proc. International Environmental Modelling and Software Congress* (2002), pp. 78–83.

[MNS08] W. Martens, F. Neven and T. Schwentick. 'Deterministic top-down tree automata: past, present, and future'. In: *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas].* Vol. 2. Texts in Logic and Games. Amsterdam University Press, 2008, pp. 505–530.

[MP43] W. S. McCulloch and W. Pitts. 'A logical calculus of the ideas immanent in nervous activity'. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133. DOI: 10.1007/BF02478259.

[MS97] A. Mateescu and A. Salomaa. 'Aspects of classical language theory'. In: *Handbook of Formal Languages.* Ed. by G. Rozenberg and A. Salomaa. Springer Berlin Heidelberg, 1997, pp. 175–251. DOI: 10.1007/978-3-642-59136-5_4.

[N68] M. Nivat. 'Transductions des langages de Chomsky'. In: *Annales de l'institut Fourier* 18.1 (1968), pp. 339–455. URL: http://eudml.org/doc/73950.

[NS02] F. Neven and T. Schwentick. 'Query automata over finite trees'. In: *Theoretical Computer Science* 275.1–2 (2002), pp. 633–674. DOI: 10.1016/S0304-3975(01)00301-2.

[P71] A. Paz. *Introduction to Probabilistic Automata.* Academic Press, 1971. DOI: 10.1016/B978-0-12-547650-8.50002-4.

[PP04] D. Perrin and J. Pin. *Infinite Words: Automata, Semigroups, Logic and Games.* Pure and Applied Mathematics. Elsevier Science, 2004.

[R63] M. O. Rabin. 'Probabilistic automata'. In: *Information and Control* 6.3 (1963), pp. 230–245.

[R69] M. O. Rabin. 'Decidability of second-order theories and automata on infinite trees'. In: *Transactions of the American Mathematical Society* 141 (1969), pp. 1–35. URL: http://www.jstor.org/stable/1995086.

[RST96] D. Ron, Y. Singer and N. Tishby. 'The power of amnesia: learning probabilistic automata with variable memory length'. In: *Machine Learning* 25 (2 1996), pp. 117–149.

[S61] M. P. Schützenberger. 'On the definition of a family of automata'. In: *Information and Control* 4.2–3 (1961), pp. 245–270. DOI: 10.1016/S0019-9958(61)80020-X.

[S88] S. Safra. 'On the complexity of ω-automata'. In: *Foundations of Computer Science (SFCS 1988)*. 1988, pp. 319–327. DOI: 10.1109/SFCS.1988.21948.

[T61] B. A. Trakhtenbrot. 'Finite automata and the logic of monadic predicates'. In: *Doklady Akademii Nauk SSSR* 140 (1961), pp. 326–329.

[TATA] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi. *Tree automata techniques and applications*. Available on: http://www.grappa.univ-lille3.fr/tata. release October, 12th 2007.

[TBG09] M. Tracol, C. Baier and M. Grösser. 'Recurrence and Transience for Probabilistic Automata'. In: *Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2009)*. Vol. 4. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2009, pp. 395–406. DOI: 10.4230/LIPIcs.FSTTCS.2009.2335.

[TW65] J. Thatcher and J. Wright. 'Generalized finite automata'. In: *Notices Amer. Math. Soc.* 12 (1965), p. 820.

[TW68] J. Thatcher and J. Wright. 'Generalized finite automata theory with an application to a decision problem of second-order logic'. In: *Mathematical systems theory* 2.1 (1968), pp. 57–81. DOI: 10.1007/BF01691346.

[VW94] M. Y. Vardi and P. Wolper. 'Reasoning about infinite computations'. In: *Information and Computation* 115.1 (1994), pp. 1–37. DOI: 10.1006/inco.1994.1092.

[W12] T. Weidner. 'Probabilistic automata and probabilistic logic'. In: *Mathematical Foundations of Computer Science (MFCS 2012)*. Vol. 7464. LNCS. Springer Berlin Heidelberg, 2012, pp. 813–824. DOI: 10.1007/978-3-642-32589-2_70.

[W14] T. Weidner. 'Probabilistic ω-regular expressions'. In: *Language and Automata Theory and Applications (LATA 2014)*. Vol. 8370. LNCS. Springer International Publishing, 2014, pp. 588–600. DOI: 10.1007/978-3-319-04921-2_48.

[W15] T. Weidner. 'Probabilistic Regular Expressions and MSO Logic on Finite Trees'. In: *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*. Vol. 45. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 503–516. DOI: 10.4230/LIPIcs.FSTTCS.2015.503.

List of Symbols

$|w|$ Length of word w

$\alpha[X \mapsto M]$ Modified assigment α mapping X to M

B_d Either B_d^D, B_d^n, or B_d^ω depending on the argument type

B_d^D Bernoulli measure on M^D for $d \in \Delta(M)$ and finite set D

B_d^n Bernoulli measure on M^n for $d \in \Delta(M)$

B_d^ω Bernoulli measure on M^ω for $d \in \Delta(M)$

B_p Either B_p^D, B_p^n, B_p^ω, or $\mathrm{B}_p^{\mathcal{P}(A)}$ depending on the argument type

B_p^D Bernoulli measure on $\{0,1\}^D$ for $p \in [0,1]$ and finite set D

B_p^n Bernoulli measure on $\{0,1\}^n$ for $p \in [0,1]$

B_p^ω Bernoulli measure on $\{0,1\}^\omega$ for $p \in [0,1]$

$\mathrm{B}_{p,E}^{\mathcal{P}(A)}$ Bernoulli measure on $\mathcal{P}(A)$ constructed with enumeration E of A

$\mathcal{B}(X)$ Borel-σ-algebra of metric space (X, d) where d is implicitly given

$\mathcal{B}(X, d)$ Borel-σ-algebra of the metric space (X, d)

$\cdot_z$ Tree concatenation

$\mathbb{1}_M$ Characteristic Function of M

$\mathrm{Cyl}_E^n(X)$ Cylinder set of the first n positions in X w.r.t. enumeration E

$\delta_q(t)$ Probability of accepting t, starting in state q

$\Delta(M)$ Set of all distributions on M

$\Delta_0(X)$ Set of all distributions and the null function on X

$\mathrm{dom}(\mathcal{A})$ Domain of structure $\mathcal{A}$

$\mathbb{E}[f]$ Expected value of function f

$\mathrm{Free}(\varphi)$ Set of free variables in MSO formula φ

$\mathrm{inner}(t)$ Set of inner positions in t

$\int f \, \mathrm{d}\mu$ Integral of function f w.r.t. measure μ

$\int f(x) \, \mu(\mathrm{d}x)$ Integral of function f w.r.t. measure μ

$\sqsubseteq$ Lexicographic Order

$\mathrm{L}(A)$ Language recognized by the automaton A

$\mathrm{L}(E)$ Language of the regular (tree) expression E

$\mathrm{L}_C(\varphi)$ Language of the MSO formula φ as subset of C

$\leq_C$ Relation on configurations — $(q, \overline{u}) \leq_C (p, \overline{v})$ if induced isomorphism is stretching

$\mathrm{leaf}(t)$ Set of leaf positions in t

$\mathrm{PMSO}(\mathcal{S})$ Set of all probabilistic MSO formulas over signature $\mathcal{S}$

PRE Set of all probabilistic regular expressions

PRTE probabilistic regular tree expressions

RE Set of all regular expressions

$\mathrm{MCAT}(\overline{w})$ Maximal common ancestor tree of $\overline{w}$

$\mathrm{MCAT}_C(\overline{w})$ Maximal common ancestor tree of $\overline{w}$ with constants C additionally included

$\mathrm{MC}(\mathcal{T}_D)$ Constraint LTL Model Checking Problem

$\mathrm{MSO}(\mathcal{S})$ Set of all MSO formulas over signature $\mathcal{S}$

$\mathbb{N}$ Natural numbers starting with 1

$\mathbb{N}_0$ Natural numbers starting with 0

$\mathcal{N}(\Sigma^\infty)$ Nivat-class of Σ^∞

$\mathcal{N}(\mathrm{T}_\Sigma)$ Nivat-class of T_Σ

$\mathcal{N}_\mathrm{D}(\mathrm{T}_\Sigma)$ Deterministic Nivat-class of T_Σ

$\|A\|$ Behaviour of the probabilistic automaton A

$\overline{R}_+$ Non-negative real numbers with ∞

$\mathrm{pos}(t)$ Set of positions in tree t

$\mathrm{pos}(w)$ Set of positions in word w

$\mathrm{pos}_A(t)$ Set of positions in t with label from A

$\mathrm{pos}_a(w)$ Set of positions of w labelled by a

$\preceq$ Prefix Order

$\mathbb{R}$ Real numbers

$\mathbb{R}_+$ Non-negative real numbers

$\mathrm{RunTypes}_n^C$ Set of all types $\mathrm{typ}(r)$ for runs r

$\mathrm{SAT}(\mathcal{T}_D)$ Constraint LTL Satisfiability Problem

$[\![\varphi]\!]_C$ Semantics of probabilistic MSO formula φ in set C

$\sigma(\mathcal{E})$ σ-algebra generated by $\mathcal{E}$

Σ^* Finite words over alphabet Σ

Σ^∞ Set of finite and infinite words over Σ

Σ^ω Infinite words over alphabet Σ

$\mathcal{T}_\Sigma$ Signature modelling finite or infinite trees

$\mathcal{W}_\Sigma$ Signature modelling finite or infinite words

$\trianglelefteq_W$ substitution order

$\sqcap$ Maximal common prefix

$\mathcal{T}_D^C$ Infinite tree with branching structure D and distinguished constants C

$\mathrm{typ}(\overline{w})$ Type of $\overline{w}$, i.e., it's $\{\preceq, \sqsubseteq, s_1, \ldots, s_{|w|}\}$-isomorphism class

$\mathrm{typ}(r)$ Type of a run $r = ((q_i, \overline{w}_i))_{i=0,\ldots,n}$ is $(q_0, \mathrm{typ}_C(\overline{w}_0, \overline{w}_n), q_n)$

$\mathrm{typ}_C(\overline{w})$ Type of $\overline{w}$ with constans from C additionally included

ε Empty word

$\widetilde{t}$ $\mathcal{T}_\Sigma$-structure associated with tree t

$\widetilde{w}$ $\mathcal{W}_\Sigma$-structure associated with word w

$A[\mathcal{X}]_\mathcal{R}$ Automaton defined as A but with no final states and Muller-condition $\mathcal{X}$

$A[X]_\mathrm{F}$ Automaton defined as A but with final states X and empty Muller-condition

B_n^C Set of propositional logic formulas build from comparison $v \sim v'$ for $v, v' \in \{x_i, y_i \mid i = 1, \ldots, n\} \cup S$

d_E Metric on $\mathcal{P}(A)$ where E is an enumeration of A

d_Σ Metric on finite or infinite words over Σ

$f(t_1, \ldots, t_n)$ Tree constructed by joining trees $t_1, \ldots, t_n$ under new root node f

$L \cdot K$ Concatenation of languages L and K

L^* Kleene-iteration of L

L^ω ω-iteration of L

$S^{\infty z}$ Infinity iteration of tree series S

$t[M \leftarrow s]$ Substitution of the subtrees at all positions from M in t by s

$t[x \leftarrow s]$ Substitution of the subtree at x in t by s

cLTL Set of all Constraint LTL formulas

Index

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

. . . . . .
(Ort, Datum)

. . . . . .
(Unterschrift)

Publikationsliste

In diesem Abschnitt befinden sich alle Veröffentlichungen mit direktem Bezug zur Dissertation.

Begutachtete Veröffentlichungen

1. T. Weidner. ‘Probabilistic automata and probabilistic logic’. In: *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012.* Vol. 7464. LNCS. Springer, 2012, pp. 813–824.

2. T. Weidner. ‘Probabilistic ω-regular expressions’. In: *Language and Automata Theory and Applications - 8th International Conference, LATA 2014.* Vol. 8370. LNCS. Springer, 2014, pp. 588–600.

3. T. Weidner. ‘Probabilistic regular expressions and MSO logic on finite trees’. In: *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015.* Vol. 45. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 503–516.

Noch unveröffentlichte Arbeiten

4. A. Kartzow and T. Weidner. ‘Model checking constraint LTL over trees’. In: *CoRR* abs/1504.06105 (2015). to be submitted.