Weighted versus Probabilistic Logics*

Benedikt Bollig and Paul Gastin

LSV, ENS Cachan, CNRS, INRIA Saclay, France
{bollig,gastin}@lsv.ens-cachan.fr

Abstract. While a mature theory around logics such as MSO, LTL, and CTL has been developed in the pure boolean setting of finite automata, weighted automata lack such a natural connection with (temporal) logic and related verification algorithms. In this paper, we will identify weighted versions of MSO and CTL that generalize the classical logics and even other quantitative extensions such as probabilistic CTL. We establish expressiveness results on our logics giving translations from weighted and probabilistic CTL into weighted MSO.

1 Introduction

Connections between logic and classical automata theory have become indispensable tools in the modeling and verification of computer systems. Usually, a logical formula φ appears as a specification, a property that a system has to fulfill, whereas an automaton $\mathcal{A}$ represents a finite-state abstraction of the system itself. Prominent examples of specification formalisms are monadic second-order (MSO) logic [35], the μ-calculus [26], and the temporal logics LTL [31] and CTL [11]. Two questions that naturally arise in this context are the *satisfiability problem* (does there exist any model of φ?) and the *model-checking problem* (do all behaviors of $\mathcal{A}$ satisfy φ?) [12].

Both logic and automata semantics give rise to a formal language that separates accepted from non-accepted behaviors. This corresponds to assigning a truth value, taken from the boolean semiring, to a behavior. When it comes to modeling and verifying quantitative systems, however, the value of a behavior is not necessarily boolean but might, e.g., be a probability of acceptance or represent a reward. Classical automata theory and logic is not suited to account for such subtleties. This led to various specialized extensions of finite automata (over finite or infinite behaviors) such as probabilistic automata [36, 33], timed automata [1], or automata with energy constraints [6], each coming with dedicated specification formalisms and approaches to related model-checking problems. In the particular case of stochastic systems, the temporal logics PCTL [24] and PCTL* [15] and corresponding model-checking techniques have been developed to reason about probabilities of events.

A generic concept of adding weights to qualitative systems is provided by the theory of weighted automata [27,28]. Unlike finite automata, which are based on

* Partially supported by projects ARCUS Île de France-Inde and ANR-06-SETIN-003 DOTS.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 18–38, 2009.

the boolean semiring, weighted automata build on more general structures such as the natural or real numbers (equipped with the usual addition and multiplication) or the probabilistic semiring. Hence, a weighted automaton associates with any possible behavior a weight beyond the usual boolean classification of "acceptance" or "non-acceptance". Automata with weights have produced a well-established theory and come, e.g., with a characterization in terms of rational expressions, which generalizes a famous counterpart in the unweighted setting. Equipped with a solid theoretical basis, weighted automata finally found their way into numerous application areas such as natural language processing and speech recognition [30], or digital image compression [14].

What is still missing in the theory of weighted automata is a satisfactory connection with logic that could lead to a general approach to related satisfiability and model-checking problems. A first step towards a logical characterization of weighted automata has been made in terms of a weighted MSO logic capturing the recognizable formal power series (i.e., the behaviors of finite weighted automata) [16, 17]. This generalizes the classical equivalence of MSO logic and finite automata [8,21]. While, however, in the qualitative setting, temporal logics such as LTL and CTL appear as fragments of MSO logic and the μ-calculus, a natural transfer of such an embedding to weighted automata is beyond the state of the art. Let us mention here some promising works that deal with this issue. In [7], Buchholz and Kemper propose valued computation-tree logic (CTL$) and corresponding model-checking algorithms for weighted Kripke structures, but do not address satisfiability and expressiveness issues. A weighted linear μ-calculus on words was defined by Meinecke, who establishes its expressive equivalence to certain ω-rational formal power series [29]. An extension towards branching structures, the identification of temporal-logic fragments, and the definition of a corresponding model-checking problem are left for future work.

Actually, only very few efforts have been made to establish a smooth connection of weighted automata with MSO and, in particular, temporal logics. We do not aim at giving final solutions to these largely open questions, but will propose a precise description of missing concepts. It is the aim of this paper to identify a weighted MSO logic as well as linear-time and branching-time logics that subsume, in a natural manner, existing quantitative logics. We will actually study the relation between our new logics and the branching-time logics PCTL and PCTL*, thus putting an emphasis on probabilistic systems.

Outline. In Section 2, we settle some notation and introduce semirings and weighted automata. Towards the end of that section, we identify probabilistic automata as a special case, which can be embedded in our framework by using a specific semiring. Sections 3 and 4 present an extended weighted MSO logic and, respectively, weighted versions of the temporal logics CTL and CTL*. They are all interpreted over unfoldings of weighted automata as introduced in Section 2 and include as special cases PCTL and PCTL*. In Section 5, we establish that our weighted temporal logic is expressible in our extended weighted MSO, transferring the well-known qualitative counterpart to the weighted case. It is also shown that the probabilistic logic PCTL can be embedded in weighted

MSO. We conclude with Section 6, in which we suggest several directions for future work.

2 Preliminaries

Words. Let Σ be an alphabet, i.e., a nonempty finite set. The set of finite words over Σ is denoted by Σ^*, the set of infinite words by Σ^ω. Moreover, we let $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$, ε denoting the empty word, and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. For $w \in \Sigma^\infty$, the length of w is denoted by $|w| \in \mathbb{N} \cup \{\omega\}$. In particular, $|\varepsilon| = 0$, and $|w| = \omega$ iff $w \in \Sigma^\omega$. Let $w = a_1 a_2 \ldots \in \Sigma^\infty$. For $i \leq |w|$, we denote $w[i] = a_1 \ldots a_i$ the prefix of w of length i, in particular, $w[0] = \varepsilon$. We denote by $\mathrm{Pref}(w)$ the set of *finite prefixes* of w. Instead of $u \in \mathrm{Pref}(v)$, we also write $u \leq v$. We write $u < v$ if, in addition, $u \neq v$. The mapping Pref is extended to sets $L \subseteq \Sigma^\infty$ in the expected manner: $\mathrm{Pref}(L) = \bigcup_{w \in L} \mathrm{Pref}(w)$. We say that $L \subseteq \Sigma^\infty$ is *prefix-closed* if $\mathrm{Pref}(L) \subseteq L$.

Semirings. A semiring is a structure $\mathbb{K} = (K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ where K is a set, $\mathbf{0}$ and $\mathbf{1}$ are constants, and $\oplus : K \times K \to K$ and $\otimes : K \times K \to K$ are binary operations, called addition and, respectively, multiplication such that $(K, \oplus, \mathbf{0})$ is a commutative monoid, $(K, \otimes, \mathbf{1})$ is a monoid, multiplication distributes over addition, and $\mathbf{0} \otimes k = k \otimes \mathbf{0} = \mathbf{0}$ for every $k \in K$. We say that $\mathbb{K}$ is *commutative* if $\otimes$ is commutative. Some popular semirings are the semiring of natural numbers $(\mathbb{N}, +, \cdot, 0, 1)$ (with the usual addition and multiplication on natural numbers), the 2-valued Boolean algebra $\mathbb{B} = (\{\mathbf{0}, \mathbf{1}\}, \vee, \wedge, \mathbf{0}, \mathbf{1})$, and the tropical semiring $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$. In this paper, we will focus on $\mathbb{P}\mathrm{rob} = (\mathbb{R}_{\geq 0}, +, \cdot, 0, 1)$, the *probabilistic semiring*, which will allow us to model probabilistic systems.[1]

The classical semirings work fine for finite trees. However, the trees that we consider might in general be infinite. We will therefore deal with infinite sums and products wrt. $\oplus$ and $\otimes$, respectively. Unfortunately, unlike finite sums and products, they do not always have a value in the semiring at hand. However, we can identify examples of infinite sums and products that are essential for our purposes and that are always defined. For arbitrary semirings $(K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, a (possibly uncountable) index set I, and $k_i \in K$ for $i \in I$, the sum $\bigoplus_{i \in I} k_i$ is defined whenever $k_i \neq \mathbf{0}$ for only finitely many i. Similarly, $\bigotimes_{i \in I} k_i$ is defined if $k_i \neq \mathbf{1}$ for finitely many i, assuming the semiring commutative or using some total order on the index set I. Considering concrete semirings such as the real numbers, a prominent infinite sum is the geometric series $\sum_{n \in \mathbb{N}} \frac{1}{2^n}$. We let its value be the limit $\lim_{n \to \infty} \sum_{n \in \mathbb{N}} \frac{1}{2^n} = 2$ of its partial sums, which is therefore defined. An example of an undefined infinite sum over the real numbers is $\sum_{n \in \mathbb{N}} \frac{1}{n}$, whose value is not in $\mathbb{R}$. We refer to the textbook [25] for a comprehensive introduction into infinite series.

If not otherwise stated, $\mathbb{K}$ will, in the following, be an arbitrary semiring $(K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ and Σ will be an alphabet.

[1] Note that $([0, 1], \max, \cdot, 0, 1)$ is sometimes considered as the probabilistic semiring as its universe restricts to probabilities. It is, however, not suitable for our purposes, as it neglects addition and, thus, does not allow one to model non-determinism.

Weighted Trees. The behavior of a non-quantitative finite-state system is often described as a (possibly infinite) tree-unfolding, whose paths constitute all possible execution sequences of the system. When we move to the quantitative setting where transitions come with weights from a semiring, then this unfolding is equipped with weights as well, which gives rise to the following definition.

Definition 1. *Let D be a nonempty finite set of* directions. *A* weighted tree *(over D, $\mathbb{K}$, and Σ) is a partial mapping $t : D^* \rightharpoonup K \times \Sigma$ such that* $\mathrm{dom}(t)$ *is prefix-closed*[2] *and $t(\varepsilon) = (\mathbf{1}, a)$ for some distinguished element a from Σ.*[3]

The set of trees over D, $\mathbb{K}$, and Σ is denoted by $\mathit{Trees}(D, \mathbb{K}, \Sigma)$. We will, however, simply write *Trees* if the parameters are understood. Let $t \in \mathit{Trees}$ be a weighted tree. It is convenient to split t into two partial mappings $\kappa_t : D^* \rightharpoonup K$ and $\ell_t : D^* \rightharpoonup \Sigma$ to extract from $t(u) = (k, a)$ the values $\kappa_t(u) = k$ and $\ell_t(u) = a$. Elements from $\mathrm{dom}(t)$ are called *nodes* of t. The empty word $\varepsilon \in \mathrm{dom}(t)$ is the *root*. A node u is a *leaf* if it is maximal in $\mathrm{dom}(t)$ for the prefix ordering, i.e., if $uD \cap \mathrm{dom}(t) = \emptyset$. If u is not a leaf, then it has some *successors*, which are nodes of the form ud with $d \in D$. The set of leaves of t is denoted by $\mathrm{Leaves}(t)$. A *branch* of t is a leaf or an infinite word whose finite prefixes are in $\mathrm{dom}(t)$. We thus define $\mathrm{Branches}(t)$ to be $\mathrm{Leaves}(t) \cup \{u \in D^\omega \mid \mathrm{Pref}(u) \subseteq \mathrm{dom}(t)\}$. Tree t is called *finite* if $\mathrm{dom}(t)$ is finite. Otherwise, it is called *infinite*. Note that we deal with unordered trees of bounded degree: we do not fix a particular order on D, and every node has at most $|D|$ successors.

We sometimes manipulate subtrees or restrictions of trees that we define now. For $u \in \mathrm{dom}(t)$, the *subtree* of t rooted at u is denoted by t_u and given by $t_u(w) = t(uw)$ for all $w \in D^+$ (and indeed $t_u(\varepsilon) = (\mathbf{1}, a)$). Given a language $L \subseteq D^\infty$, the tree $t_{|L}$ is the restriction of t to $\mathrm{dom}(t) \cap \mathrm{Pref}(L)$: $t_{|L}(u) = t(u)$ if $u \in \mathrm{dom}(t) \cap \mathrm{Pref}(L)$, and $t_{|L}(u)$ is undefined otherwise. Alternatively, one may extract a tree based on a language $L \subseteq \Sigma^\infty$ by keeping only those branches whose labeling wrt. ℓ is in $\mathrm{Pref}(L)$. Formally, we define

$$\widetilde{L} = \{u \in D^* \mid \ell_t(u[1])\ell_t(u[2]) \cdots \ell_t(u) \in \mathrm{Pref}(L)\}$$

and we are interested in $t_{|\widetilde{L}}$. When we further restrict the tree to branches that end in nodes located at directions from a set $D' \subseteq D$, then we obtain trees $t_{|L \cap D^* D'}(u)$ and $t_{|\widetilde{L} \cap D^* D'}(u)$ respectively.

Let us define a partial mapping $\widehat{\kappa} : \mathit{Trees} \rightharpoonup K$, which associates with a tree its *measure*, a weight in the semiring $\mathbb{K}$, if it exists. Intuitively, we sum over the weights of every branch. The weight of a branch, in turn, is the product of weights that are assigned to its nodes. So let, for $t \in \mathit{Trees}$,

$$\widehat{\kappa}(t) = \bigoplus_{u \in \mathrm{Branches}(t)} \bigotimes_{v \in \mathrm{Pref}(u)} \kappa_t(v) = \bigoplus_{d \in D \cap \mathrm{dom}(t)} \kappa_t(d) \otimes \widehat{\kappa}(t_d) \ .$$

[2] Let $\mathrm{dom}(t)$ be the set of words $u \in D^*$ such that $t(u)$ is defined.

[3] The value of ε will actually not be relevant so that we assume a unique value $(\mathbf{1}, a)$.

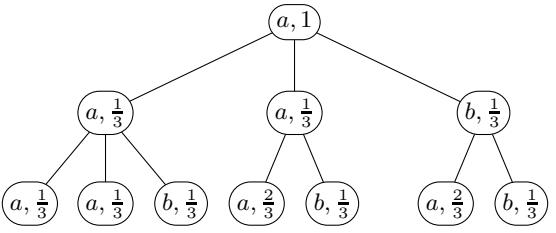


Fig. 1. A finite weighted tree over $\mathbb{P}\mathrm{rob}$, and $\{a, b\}$

Example 1. Figure 1 depicts a finite weighted tree t over $\mathbb{P}\mathrm{rob}$, and $\Sigma = \{a, b\}$. The branches of t are its leaves. We have

$$\widehat{\kappa}(t_{|\widetilde{\{aa\}}}) = \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{2}{3} = \frac{4}{9} \,.$$

Weighted Automata. In a weighted automaton, the values of a semiring that are collected along a run of the automaton are multiplied, while values of runs are summed-up.

Definition 2. *A* weighted automaton *over* $\mathbb{K}$ *and* Σ *is a quadruple* $(Q, \lambda, \mu, \gamma)$ *where* Q *is the nonempty finite set of* states*,* $\mu : \Sigma \to K^{Q \times Q}$ *is the* transition weight function*, and* $\lambda, \gamma \in Q \to K$ *provide weights for entering and leaving a state, respectively.*

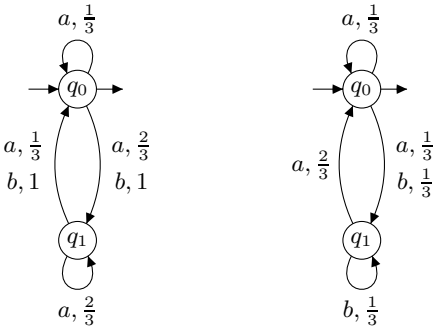
Let $\mathcal{A} = (Q, \lambda, \mu, \gamma)$ be a weighted automaton over $\mathbb{K}$ and Σ. For $a \in \Sigma$, $\mu(a)$ is a $(Q \times Q)$-matrix, and we let $\mu(a)_{p,q}$ or also $\mu(p, a, q)$ refer to its (p, q)-entry. The mapping μ uniquely extends to a monoid homomorphism $\Sigma^* \to (K^{Q \times Q}, \cdot, \mathsf{id})$ with unit matrix id where $\mathsf{id}_{p,p} = \mathbf{1}$ and $\mathsf{id}_{p,q} = \mathbf{0}$ if $p \neq q$. The semantics of $\mathcal{A}$ is given as a mapping $[\![\mathcal{A}]\!] : \Sigma^* \to K$ called a *formal power series*. Namely, for $w \in \Sigma^*$, one lets $[\![\mathcal{A}]\!](w) = \lambda \cdot \mu(w) \cdot \gamma$ with the usual matrix multiplication, considering λ as a row and γ as a column vector.

In the following, we will make two assumptions on initial and final weights:

(1) there is $q \in Q$ such that $\lambda(q) = \mathbf{1}$ and $\lambda(q') = \mathbf{0}$ for all $q' \in Q \setminus \{q\}$, and
(2) for all $q \in Q$, $\gamma(q) \in \{\mathbf{0}, \mathbf{1}\}$.

It is folklore that assuming (1) and (2) does not restrict generality, as any weighted automaton $\mathcal{A} = (Q, \lambda, \mu, \gamma)$ can be transformed into a weighted automaton $\mathcal{A}'$ that satisfies (1) and (2) and such that $[\![\mathcal{A}]\!](w) = [\![\mathcal{A}']\!](w)$ for all $w \in \Sigma^+$. Note that the transformation does not necessarily preserve the weight originally assigned to the empty word.

As we will, in the following, restrict to weighted automata that satisfy (1) and (2), we can represent $\mathcal{A}$ as the tuple (Q, q_0, μ, F) where the *initial state* $q_0 \in Q$ is the unique state q with $\lambda(q) = \mathbf{1}$, and $F = \{q \in Q \mid \gamma(q) = \mathbf{1}\}$ is the set of *final states*.



RPFA $\mathcal{A}_1$ GPFA $\mathcal{A}_2$

Fig. 2. Weighted automata over $\mathbb{P}\mathrm{rob}$

We will now define an alternative semantics and associate with a weighted automaton $\mathcal{A} = (Q, q_0, \mu, F)$ its unfolding in terms of an infinite weighted tree over $D = \Sigma \times Q$, $\mathbb{K}$, and Σ. Formally, the *unfolding* of $\mathcal{A}$, denoted by $t^{\mathcal{A}}$, is defined to be the tree $t \in \mathit{Trees}(D, \mathbb{K}, \Sigma)$ such that, for all $u \in D^*$ and $(a, q), (a', q') \in D$, the following hold: $\kappa_t((a, q)) = \mu(q_0, a, q)$, $\kappa_t(u(a, q)(a', q')) = \mu(q, a', q')$, and $\ell_t(u(a, q)) = a$. For every $w \in \Sigma^+$, we have

$$[\![\mathcal{A}]\!](w) = \widehat{\kappa}\Big(t^{\mathcal{A}}_{|\widetilde{\{w\}} \cap D^*(\Sigma \times F)}\Big) \ .$$

Example 2. Consider Figure 2 depicting weighted automata $\mathcal{A}_1$ and $\mathcal{A}_2$ over $\mathbb{P}\mathrm{rob}$ and Σ. In all cases, q_0 is both the initial state and the only final state. Missing values in $\mathcal{A}_1$ and $\mathcal{A}_2$ are supposed to be 0. For $n \in \mathbb{N}$, we have $[\![\mathcal{A}_1]\!](ab^n) = \frac{1}{3}$ if n is even, and $[\![\mathcal{A}_1]\!](ab^n) = \frac{2}{3}$ if n is odd. The set $[\![\mathcal{A}_1]\!](\Sigma^*) = \{0, \frac{1}{3}, \frac{2}{3}, 1\}$ is actually finite. This does not apply to $[\![\mathcal{A}_2]\!]$. We have, e.g., $[\![\mathcal{A}_2]\!](a) = [\![\mathcal{A}_2]\!](aa) = \frac{1}{3}$ and $[\![\mathcal{A}_2]\!](aaa) = \frac{5}{27}$. Note that $[\![\mathcal{A}_2]\!](w) = 0$ whenever w ends with the letter b. Figure 1 depicts the unfolding of $\mathcal{A}_2$ up to depth 2, i.e., $t^{\mathcal{A}_2}_{|D^2}$.

When we consider our examples to be probabilistic automata (see Definitions 3, 4), it will be evident to which extent all these values can be interpreted as probabilities of acceptance.

Probabilistic Finite Automata. There is a wide range of automata models that incorporate probabilities. We refer the reader to [36, 34] for an overview. Our generic framework of weighted automata allows us to treat many of them in a unified manner.

One basically distinguishes between *reactive* and *generative* probabilistic automata. Reactive models are input-driven: an action (from our alphabet Σ) determines a probability distribution on the set of states. The next state of an execution is then randomly drawn according to this distribution. In a generative model, the next state *and* the action are chosen according to a probability distribution so that we might call the model probability-driven.

Definition 3. *A* reactive probabilistic finite automaton (RPFA) *over* Σ *is a weighted automaton* (Q, q_0, μ, F) *over* $\mathbb{P}\mathrm{rob}$ *and* Σ *such that, for every* $q \in Q$ *and* $a \in \Sigma$*, we have* $\sum_{q' \in Q} \mu(q, a, q') \in \{0, 1\}$.[4] *In other words, we require that* $\mu(a)$ *is a stochastic matrix for every action* $a \in \Sigma$*.*

The semantics of an RPFA $\mathcal{A} = (Q, q_0, \mu, F)$ computes, for each word $w \in \Sigma^*$, a probability of acceptance as defined directly in [33]. In the following sections, we will consider mechanisms that allow us to extract from the formal power series $[\![\mathcal{A}]\!]$ a boolean language. We may, for example, be interested in the set $L = \{w \in \Sigma^* \mid [\![\mathcal{A}]\!](w) > p\}$ of words whose probabilities exceed a given threshold $p \in [0, 1]$. In general, L can be non-regular, unless p is an *isolated cut-point* of $[\![\mathcal{A}]\!]$ [33]. Moreover, it is in general undecidable if L is empty:

Theorem 1 (Rabin [33]). *The following problem is undecidable:*
INPUT*: Alphabet* Σ*;* RPFA $\mathcal{A}$ *over* Σ*;* $p \in [0, 1]$.
QUESTION*: Is there* $w \in \Sigma^*$ *such that* $[\![\mathcal{A}]\!](w) > p$ *?*

Definition 4. *A* generative probabilistic finite automaton (GPFA) *over* Σ *is a weighted automaton* (Q, q_0, μ, F) *over* $\mathbb{P}\mathrm{rob}$ *and* Σ *such that, for every* $q \in Q$*, we have* $\sum_{(a,q') \in \Sigma \times Q} \mu(q, a, q') \in \{0, 1\}$.[5]

Example 3. Let us reconsider our sample automata from Figure 2 (cf. Example 2) and let $w \in \Sigma^*$. As $\mathcal{A}_1$ is an RPFA, the weight $[\![\mathcal{A}_1]\!](w)$ can be interpreted as the probability of reaching a final state when w is used as a *scheduling policy*. E.g., $[\![\mathcal{A}_1]\!](aab) = \frac{2}{3}$. On the other hand, $[\![\mathcal{A}_2]\!](w)$ is the probability of executing w *and* ending in a final state, under the precondition that we perform $|w|$ steps. Remember that, e.g., $[\![\mathcal{A}_2]\!](a) = [\![\mathcal{A}_2]\!](aa) = \frac{1}{3}$.

3 Extended Weighted MSO

A weighted MSO logic was proposed by Droste and Gastin [16, 17] in order to extend Büchi's and Elgot's fundamental theorems [8, 21, 9] from the boolean setting to the quantitative (weighted) one. This logic was designed in order to obtain an equivalence between weighted languages (or formal power series) generated by weighted automata and those definable in this weighted MSO logic.

Other quantitative logics have been introduced and studied, e.g., PCTL [24] or PCTL* [15,10] which are probabilistic versions of the computation tree logic. These logics are evaluated on probabilistic transition systems, which are nothing

[4] Another common term for this model is simply *probabilistic finite automaton* [33]. When we neglect final states and consider the unfolding semantics rather than formal power series, then RPFA essentially correspond to the classical model of a *Markov decision process* (MDP) [32].

[5] If Σ is a singleton set, then a GPFA can be understood as a discrete-time Markov chain (DTMC). Otherwise, elements from Σ can be considered as sets of propositions that hold in the target state of a corresponding transitions. Then, we actually deal with a labeled DTMC [24].

but special instances of weighted automata as seen in Section 2. Hence, comparing weighted MSO and these logics is a natural question. It is easy to observe that these logics are *uncomparable*. Though formally correct, this answer is not very satisfactory.

Our aim is to slightly extend the weighted MSO logic in order to obtain classical quantitative logics such as PCTL and PCTL* as fragments. The crucial quantitative aspect of these logics is the probability of the set of infinite paths satisfying some linear time (LTL) property. We find it convenient to collect the set of paths in the weighted tree which is the unfolding of the probabilistic transition system. Hence, the models of our extended weighted MSO will be weighted finite or infinite trees.

The original weighted MSO logic on finite words [16] has been extended to various settings and in particular to finite trees [19] or infinite words [18] still with the aim of obtaining weighted versions of Büchi's and Elgot's fundamental theorems. These logics are also uncomparable with PCTL or PCTL*.

The key construction which is missing from all above mentioned weighted MSO logics is the possibility to transform a weighted formula into a boolean one, e.g., by using some threshold mechanism. Hence, this will be the main feature of our extension.

Our weighted MSO logic is based on a (finite) vocabulary $\mathcal{C}$ of symbols $\bowtie \in \mathcal{C}$ with $\mathrm{arity}(\bowtie) \in \mathbb{N}$. We always include negation $\neg$, disjunction $\vee$ and conjunction $\wedge$ in the vocabulary. We may also include the equality predicate $=$ and if the semiring $\mathbb{K}$ is ordered we may use the *less than* predicate $<$. Each symbol $\bowtie \in \mathcal{C}$ is given a semantics $[\![\bowtie]\!] : K^{\mathrm{arity}(\bowtie)} \to K$. To comply with the original weighted MSO, we interpret disjunction as addition $[\![\vee]\!] = \oplus$ and conjunction as multiplication $[\![\wedge]\!] = \otimes$. Depending on the semiring, the semantics of negation may be only partially defined. In any case, it is at least defined on $\mathbf{0}$ and $\mathbf{1}$ and exchanges these two values: $[\![\neg]\!](\mathbf{0}) = \mathbf{1}$ and $[\![\neg]\!](\mathbf{1}) = \mathbf{0}$. For the probabilistic semiring, we may define negation on the interval $[0, 1]$ by $[\![\neg]\!](k) = 1 - k$ or we can even make it totally defined with $[\![\neg]\!](k) = \max(0, 1 - k)$.

Definition 5. *The syntax of our weighted MSO logic is given by the grammar*

$$\varphi ::= k \mid \kappa(x) \mid P_a(x) \mid x \leq y \mid x \in X \mid \\ \mid \bowtie(\varphi_1, \ldots, \varphi_{\mathrm{arity}(\bowtie)}) \mid \exists x.\varphi \mid \exists X.\varphi \mid \forall x.\varphi \mid \forall X.\varphi$$

where $k \in K$, $a \in \Sigma$, x, y are first-order variables, X is a set variable and $\bowtie \in \mathcal{C}$. We denote by $\mathrm{MSO}(\mathbb{K}, \Sigma, \mathcal{C})$ the collection of all such formulas.

The original weighted MSO introduced in [16] is the fragment with $\mathcal{C} = \{\vee, \wedge\}$ and which does not use $\kappa(x)$. In our extension, the semantics of existential and universal quantifications will also be sums and products. In addition to the symbols $\bowtie \in \mathcal{C}$ whose semantics was already discussed, there is a new unary operator $\kappa(x)$ which gives the *weight* of the corresponding node in our models which are *weighted* trees.

Formally, we fix $t : D^* \rightharpoonup K \times \Sigma$ a weighted tree. Let $\mathcal{V}$ be a finite set of first-order and second-order variables. A $(\mathcal{V}, t)$-assignment σ is a function mapping

Table 1. Semantics of wMSO$(\mathbb{K}, \Sigma, \mathcal{C})$

$$[\![k]\!]_{\mathcal{V}}(t,\sigma) = k$$

$$[\![\kappa(x)]\!]_{\mathcal{V}}(t,\sigma) = \kappa_t(\sigma(x))$$

$$[\![P_a(x)]\!]_{\mathcal{V}}(t,\sigma) = \begin{cases} \mathbf{1} & \text{if } \ell_t(\sigma(x)) = a \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$[\![x \le y]\!]_{\mathcal{V}}(t,\sigma) = \begin{cases} \mathbf{1} & \text{if } \sigma(x) \le \sigma(y) \\ \mathbf{0} & \text{otherwise} \end{cases} \qquad \begin{array}{l} \le \text{ is the prefix} \\ \text{ordering on } \mathrm{dom}(t) \end{array}$$

$$[\![x \in X]\!]_{\mathcal{V}}(t,\sigma) = \begin{cases} \mathbf{1} & \text{if } \sigma(x) \in \sigma(X) \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$[\![\bowtie(\varphi_1,\dots,\varphi_r)]\!]_{\mathcal{V}}(t,\sigma) = [\![\bowtie]\!]([\![\varphi_1]\!]_{\mathcal{V}}(t,\sigma),\dots,[\![\varphi_r]\!]_{\mathcal{V}}(t,\sigma)) \qquad \text{if arity}(\bowtie) = r$$

$$[\![\exists x.\varphi]\!]_{\mathcal{V}}(t,\sigma) = \bigoplus_{u \in \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(t, \sigma[x \to u])$$

$$[\![\exists X.\varphi]\!]_{\mathcal{V}}(t,\sigma) = \bigoplus_{U \subseteq \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{X\}}(t, \sigma[X \to U])$$

$$[\![\forall x.\varphi]\!]_{\mathcal{V}}(t,\sigma) = \bigotimes_{u \in \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(t, \sigma[x \to u])$$

$$[\![\forall X.\varphi]\!]_{\mathcal{V}}(t,\sigma) = \bigotimes_{U \subseteq \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{X\}}(t, \sigma[X \to U])$$

first-order variables in $\mathcal{V}$ to elements of $\mathrm{dom}(t)$ and second-order variables in $\mathcal{V}$ to subsets of $\mathrm{dom}(t)$. If x is a first-order variable and $u \in \mathrm{dom}(t)$ then $\sigma[x \to u]$ is the $(\mathcal{V} \cup \{x\}, t)$-assignment which assigns x to u and acts like σ on all other variables. Similarly, $\sigma[X \to U]$ is defined for $U \subseteq \mathrm{dom}(t)$.

As usual, a pair (t, σ) where σ is a $(\mathcal{V}, t)$-assignment will be encoded using an extended alphabet $\Sigma_{\mathcal{V}} = \Sigma \times \{0,1\}^{\mathcal{V}}$. More precisely, we will write a weighted tree over $\Sigma_{\mathcal{V}}$ as a pair $(t, \sigma) : D^* \rightharpoonup K \times \Sigma_{\mathcal{V}}$ where t is the projection over $K \times \Sigma$ and σ is the projection over $\{0,1\}^{\mathcal{V}}$. Note that $\mathrm{dom}(t) = \mathrm{dom}(\sigma)$. Now, σ represents a *valid* assignment over $\mathcal{V}$ if for each first-order variable $x \in \mathcal{V}$, there is exactly one node $u \in \mathrm{dom}(\sigma)$ such that $\sigma(u)(x) = 1$.

Let now $\varphi \in$ wMSO$(\mathbb{K}, \Sigma, \mathcal{C})$. We denote as usual by $\mathrm{Free}(\varphi)$ the set of *free* variables in φ. When $\mathrm{Free}(\varphi) \subseteq \mathcal{V}$, we give in Table 1 the inductive definition of the semantics as a (partial) formal power series $[\![\varphi]\!]_{\mathcal{V}} : \mathit{Trees}(D, \mathbb{K}, \Sigma_{\mathcal{V}}) \rightharpoonup K$. We simply write $[\![\varphi]\!]$ for $[\![\varphi]\!]_{\mathrm{Free}(\varphi)}$.

Note that the semantics of a formula may be only partially defined. This may arise in particular if the semantics of some symbol in $\mathcal{C}$ is partially defined, e.g., for negation. The other difficulty is with the semantics of existential and universal quantifications.

First, if the semiring is not commutative we have to fix some order for the products of the universal quantifications. In the sequel, we will only use commutative semirings so this is not a problem. But it is also possible to deal with non

commutative products. For instance, we may use the hierarchical total ordering $\prec$ on the nodes $u \in \mathrm{dom}(t)$ for the definition of $[\![\forall x.\varphi]\!]$. With this linear order, $(\mathrm{dom}(t), \prec)$ is isomorphic to an initial segment of $(\mathbb{N}, \leq)$. Hence, the characteristic function of a subset $U \subseteq \mathrm{dom}(t)$ can be identified with a word in $\{0,1\}^{\mathrm{dom}(t)}$. So the lexicographic order on words induces a total order on the powerset of $\mathrm{dom}(t)$ which can be used to compute the product over $U \subseteq \mathrm{dom}(t)$.

Second, if the tree t is infinite, we are faced with infinite sums and infinite products. We refer to Section 2 for a discussion on when this is well-defined.

A formula is *boolean* if it only takes values in $\{\mathbf{0}, \mathbf{1}\} \subseteq K$. We call bMSO the *boolean* fragment of wMSO which consists of formulas using only constants $k \in \{\mathbf{0}, \mathbf{1}\}$ and symbols $\bowtie \in \{\neg, \wedge\}$, and which does not use $\kappa(x)$ or existential quantifications. It is easy to see that each formula $\varphi \in \mathrm{bMSO}$ takes only values in $\{\mathbf{0}, \mathbf{1}\}$ and for these formulas, the weighted semantics in $\mathbb{K}$ corresponds to the classical boolean semantics in $\mathbb{B}$. For convenience, we introduce macros for the boolean versions of disjunction and existential quantifications:

$$\varphi_1 \underline{\vee} \varphi_2 \stackrel{\text{def}}{=} \neg(\neg\varphi_1 \wedge \neg\varphi_2) \qquad \underline{\exists} x.\varphi \stackrel{\text{def}}{=} \neg\forall x.\neg\varphi \qquad \underline{\exists} X.\varphi \stackrel{\text{def}}{=} \neg\forall X.\neg\varphi$$

These *boolean* formulas are a convenient alternative to the *unambiguous* formulas introduced in [16]. We also use a boolean version of implication which is simply defined by

$$\varphi_1 \xrightarrow{+} \varphi_2 \stackrel{\text{def}}{=} \neg\varphi_1 \vee (\varphi_1 \wedge \varphi_2) \ .$$

This formula is also useful when φ_1 is boolean but not necessarily φ_2. Within a universal quantification, it allows us to compute the product of weights given by φ_2 provided φ_1 is satisfied (see Example 4 below). If φ_1 is boolean, we have

$$[\![\varphi_1 \xrightarrow{+} \varphi_2]\!]\nu(t,\sigma) = \begin{cases} [\![\varphi_2]\!]\nu(t,\sigma) & \text{if } [\![\varphi_1]\!]\nu(t,\sigma) = \mathbf{1} \\ \mathbf{1} & \text{otherwise.} \end{cases}$$

Note that the restricted form $(x \in X) \xrightarrow{+} k$ for $k \in K$ was introduced in [16,17] for the same purpose.

Example 4. To exemplify weighted MSO, we study, as models, unfoldings of the weighted automaton $\mathcal{A}_2$ over $\mathbb{P}\mathrm{rob}$ and $\Sigma = \{a, b\}$ from Figure 2 with set of states $Q = \{q_0, q_1\}$ and set of directions $D = \Sigma \times Q$. We will, thus, define trees from $\mathit{Trees}(D, \mathbb{P}\mathrm{rob}, \Sigma)$ and formulas from $\mathrm{MSO}(\mathbb{P}\mathrm{rob}, \Sigma, \{\vee, \wedge, \neg, \leq\})$. Consider the infinite weighted tree $t_1 = t^{\mathcal{A}_2}$ as well as the finite tree $t_2 = t^{\mathcal{A}_2}_{|D^2}$ (see Figure 1). We assume that roots are always labeled with $(1, a)$.

For a start, let $\varphi_1 = \exists x.(P_b(x) \wedge (\kappa(x) > 0))$. The semantics of φ_1 is the number of nodes that carry b in their labeling and have a positive weight. Though we refer to the probabilistic semiring, formula φ_1 has therefore nothing to do with a probability. The value $[\![\varphi_1]\!](t_1)$ is not even defined, as it constitutes a non-convergent infinite sum. On the other hand, $[\![\varphi_1]\!](t_2) = 4$.

Now look at $\varphi_2 = \forall x.((P_a(x) \wedge (\kappa(x) > 0)) \xrightarrow{+} \kappa(x))$ which multiplies the positive values of all a-labeled nodes. We have $[\![\varphi_2]\!](t_1) = 0$ (it is actually an

infinite product which converges to 0) and $[\![\varphi_2]\!](t_2) = \frac{4}{3^6}$. Though the semantics of φ_2 is always in the range $[0, 1]$, it can hardly be interpreted as a probability. In Section 5, we will identify a syntactical fragment of MSO that is suited to speak about probabilities and accounts for the probability space of branches (paths) of a given tree. The next property will be expressible in this fragment.

Let us first assume a boolean macro formula pathto(X, x) stating that X forms a finite branch in the tree starting at the root and ending in x. We omit the precise definition, which is similar to the formula path(x, X) given in Section 5. Now consider

$$\varphi_3 = \exists X \exists x.(\text{pathto}(X, x) \wedge P_b(x) \wedge \forall y.(y < x \xrightarrow{+} P_a(x)) \wedge \forall y.(y \in X \xrightarrow{+} \kappa(y)))$$

Indeed, $[\![\varphi_3]\!]$ computes the probability of the set of branches that contain at least one b. We have $[\![\varphi_3]\!](t_1) = 1$ (note that $[\![\varphi_3]\!](t_1)$ is an infinite sum whose partial sums converge to 1). Moreover, $[\![\varphi_3]\!](t_2) = \frac{5}{9}$. In Section 4, we will define a logic, called weighted CTL*, whose specialization to $\mathbb{P}\text{rob}$ allows us to reason about probabilities. We will show that MSO covers this fragment, by giving corresponding formulas, which actually resemble the formula φ_3: one considers the sum over the value of paths that satisfy a given boolean property.

We show now that our weighted MSO is undecidable in general. This is obtained for the probabilistic semiring $\mathbb{P}\text{rob}$ using the binary predicate *less than* even if we use unweighted words instead of weighted trees as models.

The (general) satisfiability problem is defined as follows: given a sentence $\varphi \in \text{wMSO}(\mathbb{K}, \Sigma, \mathcal{C})$, does there exist a weighted tree $t \in \mathit{Trees}(D, \mathbb{K}, \Sigma)$ such that $[\![\varphi]\!](t) \neq \mathbf{0}$.

Proposition 1. *The satisfiability problem for* $\text{wMSO}(\mathbb{P}\text{rob}, \Sigma, \{\vee, \wedge, \neg, \leq\})$ *is undecidable.*

This result is obtained with a reduction of the emptiness problem for reactive probabilistic finite automata (RPFA). Hence, it also holds if we restrict to unweighted (finite) trees or to unweighted (finite) words.

Proof. Let $\mathcal{C} = \{\vee, \wedge, \neg, \leq\}$ and let $\mathcal{A} = (Q, q_0, \mu, F)$ be a RPFA over Σ. By [16] there is a sentence $\varphi \in \text{wMSO}(\mathbb{P}\text{rob}, \Sigma, \{\vee, \wedge, \neg\})$ which does not use the unary operator $\kappa(x)$ such that for all unweighted words $w \in \Sigma^*$ we have $[\![\varphi]\!](w) = [\![\mathcal{A}]\!](w)$. We may even assume that the formula φ is existential (i.e., of the form $\exists X_1 \ldots \exists X_n.\psi$) and is syntactically restricted (see [17]).

Now, let $p \in [0, 1]$ and consider the weighted formula $p < \varphi$ using the binary predicate $\leq$. Then, for all unweighted words $w \in \Sigma^*$ we have $[\![p < \varphi]\!](w) \neq 0$ iff $[\![p < \varphi]\!](w) = 1$ iff $[\![\mathcal{A}]\!](w) > p$ iff the automaton $\mathcal{A}$ with threshold p accepts a nonempty language. By Theorem 1 we conclude that the satisfiability problem wrt. (unweighted) words is undecidable for $\text{wMSO}(\mathbb{P}\text{rob}, \Sigma, \{\vee, \wedge, \neg, \leq\})$. Since the formula φ does not use $\kappa(x)$, whether we consider weighted or unweighted words or trees does not make any difference. □

4 Weighted CTL*

We fix an *ordered* semiring $\mathbb{K}$ and a finite set *Prop* of atomic propositions. The corresponding alphabet is $\Sigma = 2^{Prop}$. As for wMSO we use a vocabulary $\mathcal{C}$ of symbols that includes $\{\neg, \vee, \wedge, \leq\}$ with the semantics given in Section 3. In our weighted CTL*, we distinguish as usual state formulas and path formulas. The path formulas are not quantitative so we call them *boolean path* formulas. The state formulas are quantitative so we use the terminology *weighted state* (or *node*) formulas. When a state formula only takes values in $\{\mathbf{0}, \mathbf{1}\}$ we call it a *boolean state* formula.

Definition 6. *The syntax of* wCTL*$(\mathbb{K}, Prop, \mathcal{C})$ *is given by the grammar*

$$\varphi ::= k \mid \kappa \mid p \mid \bowtie(\varphi_1, \ldots, \varphi_{\text{arity}(\bowtie)}) \mid \mu(\psi)$$
$$\psi ::= \varphi \mid \psi \wedge \psi \mid \neg\psi \mid \psi \,\mathsf{SU}\, \psi$$

where $p \in Prop$, $k \in K$, $\bowtie \in \mathcal{C}$, φ *is a* weighted *state formula and* ψ *is a* boolean *path formula.*

Models for wCTL*$(\mathbb{K}, Prop, \mathcal{C})$ are weighted trees $t \in Trees(D, \mathbb{K}, \Sigma)$. For weighted state formulas φ we also have to fix a node $u \in \text{dom}(t)$, and the semantics $[\![\varphi]\!](t, u) \in K$ is a value in the semiring. For boolean path formulas ψ we fix both a path $w \in \text{Branches}(t)$ and a node $u \leq w$ on this path, and the semantics defines whether the formula holds at node u on path w, denoted $t, w, u \models \psi$. The semantics are defined by induction on the formula: see Table 2 for weighted state formulas and Table 3 for boolean path formulas. We may also define the semantics of wCTL*$(\mathbb{K}, Prop, \mathcal{C})$ formulas on weighted automata by using the associated unfolding which is a weighted tree.

The semantics of $\mu(\psi)$, the *measure* of ψ, is always well-defined for finite trees. But if we consider infinite trees, it may involve infinite sums or products which are not always defined. We discuss below the special case of the probabilistic semiring for which the natural semantics of $\mu(\psi)$ on infinite trees is given by the probability measure on the sequence space. In this way, we will obtain the probabilistic logics PCTL and PCTL* as frements of wCTL*.

Table 2. Semantics of weighted state formulas in wCTL*$(\mathbb{K}, Prop, \mathcal{C})$

$$[\![k]\!](t, u) = k$$
$$[\![\kappa]\!](t, u) = \kappa_t(u)$$
$$[\![p]\!](t, u) = \begin{cases} \mathbf{1} & \text{if } p \in \ell_t(u) \\ \mathbf{0} & \text{otherwise} \end{cases}$$
$$[\![\bowtie(\varphi_1, \ldots, \varphi_r)]\!](t, u) = [\![\bowtie]\!]([\![\varphi_1]\!](t, u), \ldots, [\![\varphi_r]\!](t, u)) \qquad \text{if arity}(\bowtie) = r$$
$$[\![\mu(\psi)]\!](t, u) = \bigoplus_{w \in \text{Branches}(t) \mid t, w, u \models \psi} \; \bigotimes_{v \mid u < v \leq w} \kappa_t(v)$$

Table 3. Semantics of boolean path formulas in $\mathrm{wCTL}^*(\mathbb{K}, \mathit{Prop}, \mathcal{C})$

$t, w, u \models \varphi$	if	$[\![\varphi]\!](t, u) \neq \mathbf{0}$
$t, w, u \models \psi_1 \wedge \psi_2$	if	$t, w, u \models \psi_1$ and $t, w, u \models \psi_2$
$t, w, u \models \neg\psi$	if	$t, w, u \not\models \psi$
$t, w, u \models \psi_1 \,\mathsf{SU}\, \psi_2$	if	$\exists u < v \leq w : (t, w, v \models \psi_2$ and $\forall u < v' < v : t, w, v' \models \psi_1)$

But first we derive some useful LTL modalities. As usual, the classical *next* and *until* modalities can be obtained from the strict until:

$$\mathsf{X}\,\psi \stackrel{\text{def}}{=} \mathbf{0}\,\mathsf{SU}\,\psi \qquad \psi_1\,\mathsf{U}\,\psi_2 \stackrel{\text{def}}{=} \psi_2 \veebar (\psi_1 \wedge (\psi_1\,\mathsf{SU}\,\psi_2))$$

When dealing with probabilistic systems such as Markov chains, it is also convenient to have a *bounded* version of until. To this purpose, logics like PCTL or PCTL^* use formulas of the form $\psi_1\,\mathsf{U}^{\leq n}\,\psi_2$ for $n \in \mathbb{N}$. The semantics is that ψ_2 must hold within n time units and until then ψ_1 should hold. We may view $\mathsf{U}^{\leq n}$ or $\mathsf{SU}^{\leq n}$ as macros which are easily expressible using the next modality X.

The fragment wCTL of wCTL^* consists only of state formulas and the arbitrary $\mu(\psi)$ construct of wCTL^* is restricted to $\mu(\varphi_1\,\mathsf{SU}^{\leq n}\,\varphi_2)$ where φ_1 and φ_2 are (boolean) state formulas and $n \in \mathbb{N} \cup \{\infty\}$ (here $\mathsf{SU}^{\leq\infty}$ is the usual unbounded strict until SU).

Example 5. Figure 3 depicts a GPFA $\mathcal{A} = (Q, q_0, \mu)$ over $\Sigma = 2^{\mathit{Prop}}$ with $\mathit{Prop} = \{p, r\}$, and the initial part of its (infinite) unfolding $t = t^{\mathcal{A}}$. In both pictures, transitions and, respectively, nodes that carry the weight 0 are omitted. Moreover, inside every node $u \in D^*$ of t we have written the state reached by the corresponding path.

Consider the quantitative state formula $\varphi = \mu(1\mathsf{SU}r) \in \mathrm{wCTL}$. Table 2 allows us to compute the semantics of φ for finite trees. For instance, $[\![\varphi]\!](t_{|D^3}) = \frac{19}{27}$. Note that the boolean formula $\varphi > \frac{4}{9}$ is contained in the fragment PCTL defined below.

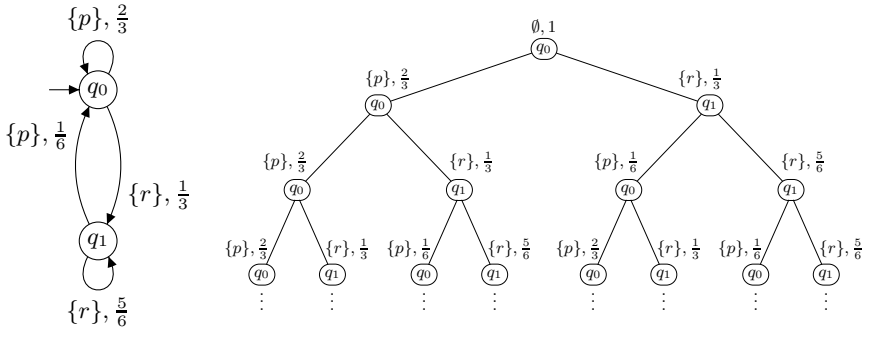


Fig. 3. GPFA $\mathcal{A} = (Q, q_0, \mu)$ over $\Sigma = 2^{\{p,r\}}$ and its unfolding $t^{\mathcal{A},q_0}$

For $n \geq 1$, consider the formula $\psi^n = \mu(\mathsf{X}^n(\mu(\mathsf{X}\,p) < \mu(\mathsf{X}\,r))) > \frac{4}{9}$ from $\mathrm{wCTL}^*(\mathbb{P}\mathrm{rob}, \mathit{Prop}, \{\neg, \wedge, \leq\})$. Formula ψ^n is neither in wCTL nor in PCTL^* (defined below). Again, Tables (2,3) allow us to compute the semantics on finite trees. For instance, the boolean formula $\mu(\mathsf{X}\,p) < \mu(\mathsf{X}\,r)$ holds precisely in state q_1. We can check that for $n < m$ we have $[\![\psi^n]\!](t_{|D^m}) = 1$ iff $n \geq 2$.

Finally, using the semantics from Table 2, it is not clear how to compute $[\![\varphi]\!](t)$ since we have to deal with an infinite sum of infinite products. A possibility is to set $[\![\varphi]\!](t) = 0$ since the infinite products all converge to 0. But this is not the desired semantics which should measure the probability that r eventually holds. Hence, we should obtain $[\![\varphi]\!](t) = 1$.

Therefore, we extend below the semantics to infinite trees in a suitable way. For finite trees, however, both semantics coincide.

We restrict to the probabilistic semiring $\mathbb{P}\mathrm{rob}$ and to trees that are unfoldings of generative probabilistic finite automata (GPFA). More precisely, we consider a GPFA $\mathcal{A} = (Q, \mu)$ over $\Sigma = 2^{\mathit{Prop}}$ (the initial state will be fixed later and final states are irrelevant in the following). Usually, atomic propositions are associated with states. Here they are associated with transitions which is a minor difference as already noticed in Definition 4. We also assume that there are no deadlock, i.e., $\sum_{(a,q') \in \Sigma \times Q} \mu(q, a, q') = 1$ for all $q \in Q$. This is not a restriction since we may always add a sink state.

Let $t^{\mathcal{A},q} \in \mathit{Trees}(D, \mathbb{P}\mathrm{rob}, \Sigma)$ be the full tree over $D = \Sigma \times Q$ obtained by unfolding the GPFA $\mathcal{A}$ with initial state $q \in Q$ (see definition in Section 2). Since $\mathcal{A}$ is fixed, we simply write $t^{\mathcal{A},q} = t^q$. As they arise from a *finite state* probabilistic system $\mathcal{A}$ the trees t^q are regular. More precisely, for $q \in Q$ and $u \in D^*$, we define $\mathrm{last}(q, u) = q$ if $u = \varepsilon$ and $\mathrm{last}(q, u) = q'$ if $u \in D^*(\Sigma \times \{q'\})$. It is easy to check that the subtree t^q_u of t^q rooted at u is in fact $t^{\mathrm{last}(q,u)}$.

The *sequence probability space* associated with $\mathcal{A}$ and initial state $q \in Q$ is $(D^\omega, \mathfrak{B}, \mathrm{prob}^q)$ where the Borel field $\mathfrak{B}$ is generated by the basic cylinders sets uD^ω with $u \in D^*$ and prob^q is the unique probability measure such that for a basic cylinder uD^ω we obtain the probability of the finite path described by u: if $u = (a_1, q_1)(a_2, q_2) \cdots (a_n, q_n)$ and with $q_0 = q$ then

$$\mathrm{prob}^q(uD^\omega) = \prod_{i=1}^{n} \mu(q_{i-1}, a_i, q_i) = \prod_{v \in \mathrm{Pref}(u)} \kappa_{t^q}(u) \ .$$

Any ω-regular set $L \subseteq D^\omega$ is measurable [37]. More precisely, any ω-regular language is a finite boolean combination of languages at the second level of the Borel hierarchy. This is a consequence of McNaughton's theorem showing that ω-regular languages can be accepted by *deterministic* Muller automata. Hence, they are finite boolean combinations of languages accepted by *deterministic* Büchi automata. Let F be the accepting set of states of a *deterministic* Büchi automaton $\mathcal{B}$ and for $n \in \mathbb{N}$, let L_n be the set of *finite* words whose run on $\mathcal{B}$ visits F at least n times. Then the language accepted by $\mathcal{B}$ is

$$\bigcap_{n\geq 0} \bigcup_{w\in L_n} wD^\omega$$

By the very definition of *strict until*, one sees that every LTL formula ψ is first-order definable, hence defines an ω-regular language $\mathcal{L}(\psi)$ by [9]. Antoher argument is to use classical translations from LTL formulas to Büchi automata. It follows that $\mathcal{L}(\psi)$ is measurable and $\mathrm{prob}^q(\mathcal{L}(\psi))$ is well-defined for LTL formulas ψ. Hence, we will be able to define the semantics of $\mu(\psi)$ using the probability measure of the sequence space.

Formally, let $q \in Q$, $u \in D^*$ and ψ be a boolean path formula. We define

$$\mathcal{L}^q_u(\psi) = \{w \in uD^\omega \mid t^q, w, u \models \psi\}$$

and

$$[\![\mu(\psi)]\!](t^q, u) = \mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L}^q_u(\psi))$$

where $u^{-1}L = \{v \in D^\infty \mid uv \in L\}$ is the left quotient of $L \subseteq D^\infty$ by $u \in D^*$. Recall that $\mathrm{last}(q, \varepsilon) = q$ and $\mathrm{last}(q, u) = q'$ if $u \in D^*(\Sigma \times \{q'\})$.

Using the remarks above, we can show by induction on the state formulas φ and the boolean path formulas ψ in $\mathrm{wCTL}^*(\mathbb{P}\mathrm{rob}, \mathit{Prop}, \mathcal{C})$ that for all $q \in Q$ and $u \in D^*$, $[\![\varphi]\!](t^q, u)$ only depends on $\mathrm{last}(q, u)$, and $\mathcal{L}^q_u(\psi)$ is measurable. Hence, the semantics of $\mu(\psi)$ is well-defined.

Example 6. Let us continue the discussion started in Example 5. Using the probabilistic semantics defined above for infinite trees, we obtain now

$$[\![\mu(1 \,\mathsf{SU}\, r)]\!](t, \varepsilon) = \mathrm{prob}^{q_0}(D^*(\{\{r\}, \{p, r\}\} \times Q)D^\omega) = 1$$

which is the probability that r eventually holds.

The probabilistic computation tree logic PCTL* [15] is a *boolean* fragment of $\mathrm{wCTL}^*(\mathbb{P}\mathrm{rob}, \mathit{Prop}, \{\neg, \wedge, \leq\})$ using the semantics defined above for $\mu(\psi)$. The restriction is on *state* formulas which

- use only constants $k \in \{\mathbf{0}, \mathbf{1}\}$ and symbols $\bowtie \in \{\neg, \wedge\}$,
- which do not use κ,
- and use $\mu(\psi)$ only with comparisons of the form $(\mu(\psi) \bowtie p)$ with $\bowtie \in \{\geq, >\}$, $p \in [0, 1]$, and ψ a path formula.

A further restriction is PCTL introduced in [24] where only state formulas are considered. Here the path formulas ψ used in $(\mu(\psi) \bowtie p)$ are restricted to be of the form $\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2$ where φ_1, φ_2 are boolean state formulas and $n \in \mathbb{N} \cup \{\infty\}$. Hence, PCTL is also a fragment of wCTL. Note that, if φ_1, φ_2 are boolean state formulas and $n \in \mathbb{N} \cup \{\infty\}$ then

$$[\![\mu(\varphi_1 \,\mathsf{U}^{\leq n}\, \varphi_2)]\!] = [\![\varphi_2 \vee (\varphi_1 \wedge \neg\varphi_2 \wedge \mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2))]\!] \,.$$

Table 4. Translation in bMSO of boolean path formulas in wCTL*

$$\underline{\varphi}(x,X) = (\overline{\varphi}(x) \neq \mathbf{0})$$
$$\underline{\psi_1 \wedge \psi_2}(x,X) = \underline{\psi_1}(x,X) \wedge \underline{\psi_2}(x,X)$$
$$\underline{\neg\psi}(x,X) = \neg\underline{\psi}(x,X)$$
$$\underline{\psi_1 \,\mathsf{SU}\, \psi_2}(x,X) = \underline{\exists} z.(z \in X \wedge x < z \wedge \underline{\psi_2}(z,X) \wedge \forall y.((x < y < z) \xrightarrow{+} \underline{\psi_1}(y,X)))$$

5 wCTL* Is a Fragment of wMSO

In this section, we will give a translation from wCTL* formulas to weighted MSO formulas.

We start with path formulas ψ in wCTL*. Implicitely, such a formula has two free variables, the path (branch) on which the formula is evaluated and the current node on this path. Naturally, the current node is a first-order variable and the path in the tree can be described by the set variable consisting of all nodes on this branch. So we associate with each path formula $\psi \in \text{wCTL}^*(\mathbb{K}, \mathit{Prop}, \mathcal{C})$ a boolean MSO formula $\underline{\psi}(x,X) \in \text{bMSO}(\mathbb{K}, \Sigma, \mathcal{C})$. The definition by induction on the formula is given in Table 4. In this definition, we assume that the interpretation of X is indeed a path. We make sure to define *boolean* formulas by using $\underline{\exists}$, $\underline{\vee}$ and $\xrightarrow{+}$ which were defined in Section 3.

Next, we turn to (weighted) state formulas $\varphi \in \text{wCTL}^*$. Here, the only implicit free variable is the current node. Hence, we associate with each state formula $\varphi \in \text{wCTL}^*(\mathbb{K}, \mathit{Prop}, \mathcal{C})$ a weighted MSO formula $\overline{\varphi}(x) \in \text{bMSO}(\mathbb{K}, \Sigma, \mathcal{C})$. The translation, which is indeed by induction on the formula, is given in Table 5. The *boolean* formula $\text{path}(x,X)$ states that X is a *maximal* path in the tree starting from node x. To this aim, we use the boolean formula $y \lessdot z$ which holds if z is a successor (son) of y in the tree:

$$y \lessdot z \stackrel{\text{def}}{=} y < z \wedge \forall z'.\neg(y < z' < z) \ .$$

The translation of $\mu(\psi)$ given in Table 5 is valid when the models are *finite* trees. For infinite trees, the set of paths is usually infinite and the semantics of $\mu(\psi)$ would involve infinite products and sums that are not necessarily defined.

As explained in Section 4, it is crucial for applications to probabilistic systems to be able to deal with infinite trees that arise as unfoldings of GPFA's. So we give below a translation of $\mu(\psi)$ to wMSO such that the infinite sums and products involved in the semantics are always well-defined.

We deal with the fragment wCTL where the path formulas are restricted to be of the form $\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2$ where φ_1 and φ_2 are boolean state formulas and $n \in \mathbb{N} \cup \{\infty\}$. The translation in wMSO of $\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)$ is given in Table 6. The boolean formula $\text{path}^{\leq \infty}(x,X)$ states that X is a path starting from x but do not impose that X is maximal (contrary to the definition in Table 5). When $n \in \mathbb{N}$, the boolean formula $\text{path}^{\leq n}(x,X)$ requires in addition that the path X is of length at most n. Assuming that X is a path starting from x, the boolean

Table 5. Translation in bMSO of boolean path formulas in wCTL*

$$
\begin{aligned}
\overline{k}(x) &= k \\
\overline{\kappa}(x) &= \kappa(x) \\
\overline{p}(x) &= \neg \bigwedge_{a \in \Sigma \mid p \in a} \neg P_a(x) \\
\overline{\bowtie(\varphi_1, \dots, \varphi_r)}(x) &= \bowtie(\overline{\varphi_1}(x), \dots, \overline{\varphi_r}(x)) \qquad \text{if arity}(\bowtie) = r \\
\overline{\mu(\psi)}(x) &= \exists X.(\mathrm{path}(x, X) \wedge \underline{\psi}(x, X) \wedge \xi(x, X) \\
\mathrm{path}(x, X) &= x \in X \\
&\quad \wedge \forall z.(z \in X \xrightarrow{+} (z = x \mathbin{\underline{\vee}} \underline{\exists} y.(y \in X \wedge y \lessdot z))) \\
&\quad \wedge \neg \underline{\exists}\, y, z, z' \in X.(y \lessdot z \wedge y \lessdot z' \wedge z \neq z') \\
&\quad \wedge \forall y.(\ (y \in X \wedge \underline{\exists}\, z.(y < z)) \xrightarrow{+} \underline{\exists}\, z.(z \in X \wedge y < z)\ ) \\
\xi(x, X) &= \forall y.((y \in X \wedge x < y) \xrightarrow{+} \kappa(y))
\end{aligned}
$$

formula $\underline{\psi}(x, X)$ states that the path satisfies $\varphi_1 \,\mathsf{SU}\, \varphi_2$ and is minimal with this property. In particular, such a path must be finite, even if $n = \infty$, since the formula $\neg(\mathbf{0} \,\mathsf{SU}\, \mathbf{1})$ means that there is no next state. Finally, the formula $\xi(x, X)$ computes the probability of the path.

Proposition 2. *Let $\mathcal{A} = (Q, \mu)$ be a* GPFA*, $q \in Q$. Let t^q be the tree unfolding of $\mathcal{A}$ starting from state q and let $u \in D^* = \mathrm{dom}(t^q)$ be a node. Let $\varphi_1, \varphi_2 \in$* wCTL$(\mathbb{P}\mathrm{rob}, \mathit{Prop}, \mathcal{C})$ *be boolean state formulas and $n \in \mathbb{N} \cup \{\infty\}$. Then,*

$$
\begin{aligned}
\left[\!\left[\overline{\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)} \right]\!\right](t^q, u) &= \mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L}_u^q(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)) \\
&= [\![\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)]\!](t^q, u)
\end{aligned}
$$

The tree t^q is infinite. Hence to prove this proposition we have to make sense of the infinite products and sums that arise from the semantics of the wMSO formula $\overline{\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)}$ given in Table 6. So let $X \subseteq D^* = \mathrm{dom}(t^q)$. By definition of $\underline{\exists}$, $\underline{\vee}$ and $\xrightarrow{+}$, the semantics of the boolean formula $\mathrm{path}^{\leq n}(u, X) \wedge \underline{\psi}(u, X)$ uses (infinite) products of boolean values $\mathbf{0}$ and $\mathbf{1}$. Naturally, we define such (infinite) products to be $\mathbf{0}$ if at least one factor is $\mathbf{0}$ and to be $\mathbf{1}$ otherwise. Hence, the difficulty is only to make sense of the (infinite) sum associated with $\exists X$ and of the (infinite) product used for the semantics of ξ.

Proof (of Prop. 2). We use the notation introduced above. We simply denote by $\mathcal{L}$ the language $\mathcal{L}_u^q(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)$ introduced in Section 4:

$$
\mathcal{L} = \{w \in uD^\omega \mid t^q, w, u \models \varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2\}\ .
$$

In this special case, the probability measure $\mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L})$ is easy to compute. To this aim, we introduce the language

$$\mathcal{K} = \{w \in uD^+ \mid t^q, w, u \models (\varphi_1 \wedge \neg\varphi_2)\ \mathsf{SU}^{\leq n}\ (\varphi_2 \wedge \neg(\mathbf{0}\ \mathsf{SU}\ \mathbf{1}))\}$$

We can easily check that $\mathcal{K}$ is prefix-free: $w, w' \in \mathcal{K}$ and $w \leq w'$ implies $w = w'$. Moreover,

$$\mathcal{L} = \biguplus_{w \in \mathcal{K}} wD^\omega$$

and this countable union is disjoint so that

$$\mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L}) = \sum_{w \in \mathcal{K}} \mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}wD^\omega) = \sum_{w \in \mathcal{K}} \prod_{u < v \leq w} \kappa_{t^q}(v)\ .$$

For each $w \in uD^+$, let $X_w = \{v \in D^* \mid u \leq v \leq w\}$. Next, define the set $\mathfrak{X} = \{X_w \subseteq D^* \mid w \in \mathcal{K}\}$. One can check that $\mathfrak{X}$ is precisely the set of subsets $X \subseteq D^* = \mathrm{dom}(t^q)$ such that the formula $\mathrm{path}^{\leq n}(u, X) \wedge \underline{\psi}(u, X)$ holds:

$$[\![\mathrm{path}^{\leq n} \wedge \underline{\psi}]\!](t^q, u, X) = \begin{cases} \mathbf{1} & \text{if } X \in \mathfrak{X} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Note that the infinite product used in the semantics of $[\![\xi]\!](u, X)$ is always well-defined. Either it has only finitely many factors different from $\mathbf{1}$ (which is in particular the case when X is finite) or it converges to $\mathbf{0}$. For each $w \in \mathcal{K}$, $[\![\xi]\!](u, X_w)$ computes the probability of path X_w:

$$[\![\xi]\!](t^q, u, X_w) = \prod_{u < v \leq w} \kappa_{t^q}(v)\ .$$

For sets $X \notin \mathfrak{X}$, we have $[\![\mathrm{path}^{\leq n} \wedge \underline{\psi} \wedge \xi]\!](t^q, u, X) = \mathbf{0}$ and we obtain

$$[\![\mathrm{path}^{\leq n} \wedge \underline{\psi} \wedge \xi]\!](t^q, u, X) = \begin{cases} [\![\xi]\!](t^q, u, X) & \text{if } X \in \mathfrak{X} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Removing $\mathbf{0}$ terms in an infinite sum, we obtain

$$\begin{aligned} \left[\!\!\left[\overline{\mu(\varphi_1\ \mathsf{SU}^{\leq n}\ \varphi_2)} \right]\!\!\right](t^q, u) &= [\![\exists X.(\mathrm{path}^{\leq n} \wedge \underline{\psi} \wedge \xi)]\!](t^q, u) \\ &= \sum_{X \in \mathfrak{X}} [\![\xi]\!](t^q, u, X) \\ &= \sum_{w \in \mathcal{K}} \prod_{u < v \leq w} \kappa_{t^q}(v) \\ &= \mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L})\ . \end{aligned}$$

If $n \in \mathbb{N}$ then the sets $\mathcal{K}$ and $\mathfrak{X}$ are finite and the sums above are finite. When $n = \infty$, i.e., for the natural unbounded strict until, the sets $\mathcal{K}$ and $\mathfrak{X}$ may be infinite. For instance, this is the case with formula $p\,\mathsf{SU}\,r$ evaluated on the GPFA of Figure 3 starting from state q_0 where the sets $\mathcal{K}$ and $\mathfrak{X}$ corresponds to the infinite set of paths $q_0^* q_1$. But, as a consequence of the equalities above, the infinite sums over $\mathcal{K}$ and $\mathfrak{X}$ are well-defined with value in $[0, 1]$. □

Table 6. Translation in bMSO of $\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2) \in$ wCTL for $n \in \mathbb{N} \cup \{\infty\}$

$$\overline{\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)}(x) = \exists X.(\mathrm{path}^{\leq n}(x,X) \wedge \underline{\psi}(x,X) \wedge \xi(x,X))$$

$$\begin{aligned}\mathrm{path}^{\leq \infty}(x,X) &= x \in X \\ &\quad \wedge \forall z.(z \in X \xrightarrow{+} (z = x \,\underline{\vee}\, \underline{\exists} y.(y \in X \wedge y \lessdot z))) \\ &\quad \wedge \neg \underline{\exists} y,z,z' \in X.(y \lessdot z \wedge y \lessdot z' \wedge z \neq z')\end{aligned}$$

$$\begin{aligned}\text{if } n \in \mathbb{N}, \quad \mathrm{path}^{\leq n}(x,X) &= \mathrm{path}^{\leq \infty}(x,X) \wedge \neg \underline{\exists} x_0 \ldots \underline{\exists} x_n. \\ &\qquad (x_0 \in X \wedge \cdots \wedge x_n \in X \wedge x < x_0 < x_1 < \cdots < x_n)\end{aligned}$$

$$\psi = (\varphi_1 \wedge \neg\varphi_2) \,\mathsf{SU}\, (\varphi_2 \wedge \neg(\mathbf{0} \,\mathsf{SU}\, \mathbf{1}))$$

$$\xi(x,X) = \forall y.((y \in X \wedge x < y) \xrightarrow{+} \kappa(y))$$

6 Conclusion and Open Problems

In this paper, we have introduced wMSO, a weighted version of classical MSO logic. It is interpreted over weighted trees, which naturally appear as unfoldings of weighted automata. We showed that the satisfiability problem for wMSO is undecidable. We then defined wCTL and wCTL* over weighted trees and gave transformations of these logics into wMSO. For the probabilistic interpretation of the path-quantifier operator $\mu(\psi)$, we restricted to a transformation of wCTL into wMSO formulas.

Let us mention some directions for future work. We need to identify fragments of our logic that come with a decidable satisfiability problem. A natural further step is to tackle the model-checking problem: given a weighted formula φ and a weighted automaton $\mathcal{A}$, does $[\![\varphi]\!](t^{\mathcal{A}}) \neq \mathbf{0}$ hold? To find a solution, we might borrow techniques used in the probabilistic setting for PCTL*. Moreover, the translation of wCTL* into wMSO remains to be done. For the probabilistic semantics, such a translation might be based on techniques from [13] developed for checking linear-time properties of probabilistic systems (see also [5] for an overview).

It would be worthwhile to add the notion of a *scheduler* to wMSO to consider unfoldings of (partially observable) MDPs or probabilistic Büchi automata [3,4], which are essentially RPFA with a Büchi acceptance condition.

The *expectation semiring*, defined in [20], combines probabilities with expected rewards. A transfer of our logics to this specific structure (possibly extended by a discount operator) could provide a generic framework for reward models as considered, e.g., in [23, 2].

A weighted μ-calculus has been defined in [22] to be interpreted over quantitative Kripke structures. For the design of a weighted μ-calculus over branching structures, one might benefit from ideas that led to a weighted μ-calculus over words [29].

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
2. Andova, S., Hermanns, H., Katoen, J.P.: Discrete-time rewards model-checked. In: Larsen, K.G., Niebert, P. (eds.) FORMATS 2003. LNCS, vol. 2791, pp. 88–104. Springer, Heidelberg (2004)
3. Baier, C., Bertrand, N., Größer, M.: On decision problems for probabilistic Büchi automata. In: Amadio, R. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 287–301. Springer, Heidelberg (2008)
4. Baier, C., Größer, M.: Recognizing ω-regular languages with probabilistic automata. In: Proceedings of LICS 2005, pp. 137–146. IEEE Computer Society Press, Los Alamitos (2005)
5. Bollig, B., Leucker, M.: Verifying qualitative properties of probabilistic programs. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 124–146. Springer, Heidelberg (2004)
6. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 33–47. Springer, Heidelberg (2008)
7. Buchholz, P., Kemper, P.: Model checking for a class of weighted automata. Discrete Event Dynamic Systems (to appear, 2009)
8. Büchi, J.R.: Weak second-order arithmetic and finite automata. Z. Math. Logik Grundlagen Math. 6, 66–92 (1960)
9. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Proc. of the International Congress on Logic, Methodology and Philosophy, pp. 1–11. Standford University Press (1962)
10. Ciesinski, F., Größer, M.: On probabilistic computation tree logic. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 147–188. Springer, Heidelberg (2004)
11. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982)
12. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. The MIT Press, Cambridge (1999)
13. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. Journal of the ACM 42(4), 857–907 (1995)
14. Culik, K., Kari, J.: Image compression using weighted finite automata. Computer and Graphics 17(3), 305–313 (1993)
15. de Alfaro, L.: Formal verification of probabilistic systems. Technical report, Stanford University, PhD thesis (1998)
16. Droste, M., Gastin, P.: Weighted automata and weighted logics. Theoretical Computer Science 380(1-2), 69–86 (2007); Special issue of ICALP 2005
17. Droste, M., Gastin, P.: Weighted automata and weighted logics. In: Kuich, W., Vogler, H., Droste, M. (eds.) Handbook of Weighted Automata. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg (to appear, 2009)
18. Droste, M., Rahonis, G.: Weighted automata and weighted logics with discounting. In: Holub, J., Žďárek, J. (eds.) CIAA 2007. LNCS, vol. 4783, pp. 73–84. Springer, Heidelberg (2007)

19. Droste, M., Vogler, H.: Weighted tree automata and weighted logics. Theoretical Computer Science 366(3), 228–247 (2006)
20. Eisner, J.: Expectation semirings: Flexible EM for learning finite-state transducers. In: Proceedings of the ESSLLI workshop on finite-state methods in NLP (2001)
21. Elgot, C.C.: Decision problems of finite automata design and related arithmetics. Trans. Amer. Math. Soc. 98, 21–52 (1961)
22. Fischer, D., Grädel, E., Kaiser, L.: Model checking games for the quantitative μ-calculus. Theory of Computing Systems (2009); Special Issue of STACS 2008
23. Größer, M., Norman, G., Baier, C., Ciesinski, F., Kwiatkowska, M., Parker, D.: On reduction criteria for probabilistic reward models. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 309–320. Springer, Heidelberg (2006)
24. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
25. Knopp, K.: Theory and Application of Infinite Series. Dover Publications, New York (1990); Republication of the second English edn. (1951)
26. Kozen, D.: Results on the propositional μ-calculus. Theoretical Computer Science 27, 333–354 (1983)
27. Kuich, W., Salomaa, A.: Semirings, Automata and Languages. Springer, Heidelberg (1985)
28. Kuich, W., Vogler, H., Droste, M. (eds.): Handbook of Weighted Automata. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg (2009)
29. Meinecke, I.: A weighted μ-calculus on words. In: Diekert, V., Nowotka, D. (eds.) DLT 2009. LNCS, vol. 5583. Springer, Heidelberg (2009)
30. Mohri, M.: Finite-state transducers in language and speech processing. Computational Linguistics 23(2), 269–311 (1997)
31. Pnueli, A.: The temporal logic of programs. In: Proceedings of FOCS 1977, pp. 46–57. IEEE Computer Society Press, Los Alamitos (1977)
32. Puterman, M.L.: Markov Decision Processes. John Wiley & Sons, Inc., New York (1994)
33. Rabin, M.O.: Probabilistic automata. Information and Control 6, 230–245 (1963)
34. Segala, R.: Probability and nondeterminism in operational models of concurrency. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 64–78. Springer, Heidelberg (2006)
35. Thomas, W.: Languages, automata and logic. In: Salomaa, A., Rozenberg, G. (eds.) Handbook of Formal Languages. Beyond Words, vol. 3, pp. 389–455. Springer, Heidelberg (1997)
36. van Glabbeek, R.J., Smolka, S.A., Steffen, B.: Reactive, generative and stratified models of probabilistic processes. Information and Computation 121(1), 59–80 (1995)
37. Vardi, M.Y.: Automatic verification of probabilistic concurrent finite-state programs. In: Proceedings of FOCS 1985, pp. 327–338. IEEE, Los Alamitos (1985)