Weight Monitoring with Linear Temporal Logic: Complexity and Decidability *

Christel Baier Joachim Klein Sascha Klüppelholz Sascha Wunderlich

Institute for Theoretical Computer Science
Technische Universität Dresden, Germany
{baier,klein,klueppel,wunder}@tcs.inf.tu-dresden.de

Abstract

Many important performance and reliability measures can be formalized as the accumulated values of weight functions. In this paper, we introduce an extension of linear time logic including past (LTL) with new operators that impose constraints on the accumulated weight along path fragments. The fragments are characterized by regular conditions formalized by deterministic finite automata (monitor DFA). This new logic covers properties expressible by several recently proposed formalisms. We study the model-checking problem for weighted transition systems, Markov chains and Markov decision processes with rational weights. While the general problem is undecidable, we provide algorithms and sharp complexity bounds for several sublogics that arise by restricting the monitoring DFA.

Categories and Subject Descriptors F.4.1 [*Mathematical logic and formal languages*]: Mathematical logic—Temporal logic

General Terms Theory

Keywords LTL, MDP, Markov chain, transition system, accumulation, weights, rewards, model checking, finite automata, monitor

1. Introduction

In many application scenarios weight accumulation occurs rather naturally. One example is the total win or loss of a share at the stock market over one day. However, not only fixed periods of time are of interest. Formalizing more general time spans is often necessary, for example in between certain events, e.g., when considering the average CPU load within a specific computation phase. Many performance and reliability measures can be formalized using automata models with weight functions for the states or transitions. Resource requirements (e.g., bandwidth, energy consumption) and other quantitative system properties (e.g., the number of service-level violations) are then formally modeled as accumulated weights of path fragments.

Various models, logics and specification formalisms for weighted structures and accumulated weights have been proposed in the literature. For the analysis of Markovian models, the traditional approaches mainly concentrate on branching time logics with cost- or reward-bounded temporal modalities and state formulas for reasoning about total expected costs or long-run averages, see e.g. [2, 4, 17, 20]. This work has mainly focused on *non-negative* weight functions, called *reward functions*. In the case of models with two or more weight functions, the properties that can be specified in such branching-time logics are mostly Boolean combinations of formulas, each of them referring to a *single* reward function. Although nested state formulas can impose constraints for different weight functions, these logics are not adequate to express, e.g., a reachability constraint with bounds for the accumulated value of two reward functions. Approaches for multi-objective reasoning in Markovian models with nondeterminism (MDPs) [12, 19, 20] focus on the task to synthesize schedulers satisfying multiple constraints on the probabilities of ω-regular events or expected (total) accumulated rewards.

There is a recent trend to study logics and algorithms for reasoning about properties on the accumulated values of multiple weight function that might have both positive and negative values. Such weight functions appear naturally when modeling, e.g., the amount of available energy in a battery or the market trend of stocks that might vary (increase or decrease) over time. Conditions on the relation between the accumulated values of different weight functions can be useful, e.g., for the analysis of load balancing algorithms for multi-core systems that might trigger a migration in case the load of one core is two times larger than the average load of the cores. Another example is constraints on the cost/utility ratio. Ratio objectives for weighted MDPs have been studied for example in [28] where the goal is to synthesize a scheduler for a given weighted MDP that maximizes or minimizes the average ratio payoff of two integer weight functions. Quantitative objectives for efficient synthesis have also been proposed in [7, 13]. An extension of linear time and branching time logic with prefix-accumulation assertions, which sum the weights from the start of the computation, is proposed in [8]. Similarily, variants with assertions on the long-run (mean-payoff) accumulation are considered in [8, 18, 26]. Decidability results for special property types $\psi \wedge \varphi$ where ψ is a condition on the accumulated value of a single weight function and φ an ω-regular path property have been established for various types of weighted game structures, e.g., for energy and mean-payoff games in [1, 10, 11].

* The authors are supported by the DFG through the collaborative research centre HAEC (SFB 912), the cluster of excellence cfAED (center for Advancing Electronics Dresden), the Graduiertenkolleg 1763 (QuantLA), and the DFG/NWO-project ROCKS, the ESF young researcher group IMData 100098198 and the EU-FP-7 grant 295261 (MEALS).

CSL-LICS 2014, July 14–18, 2014, Vienna, Austria.
Copyright © 2014 ACM 978-1-4503-2886-9…$15.00.
http://dx.doi.org/10.1145/2603088.2603162

We introduce a temporal logical framework based on linear temporal logic (LTL) with the standard future and past temporal modalities and two new operators $\ominus^{\mathcal{A}}$ and $\oplus^{\mathcal{A}}$. The latter impose constraints on the accumulated weight of path fragments π satisfying a given regular condition formalized by a deterministic finite automaton (DFA) $\mathcal{A}$, possibly under side constraints formalized by nested formulas for the first and the last position of π. Formulas are interpreted over the paths of weighted structures. We consider finite-state Markov decision processes with multiple weight functions, briefly called WMDPs. MDPs are a standard mathematical model with nondeterministic and probabilistic choices. For linear-time specifications, the probabilistic model-checking (PMC) problem for MDPs denotes the task to compute the maximal or minimal probabilities of a specification, where extrema are taken when ranging over all resolutions of the nondeterministic choices. Weighted transition systems (WTSs) and weighted Markov chains (WMCs) can be seen as degenerated WMDPs without nondeterminism or probabilism, respectively.

Four different classes of automata are considered here. Window DFA simply impose a restriction on the length of words. They can be used to formalize, e.g., constraints on the load of a processor in the past few clock cycles or the chart-development of a stock within one day. The fixed window properties studied in [14] for non-probabilistic game structures are expressible in our logic with window DFA. More flexibility is provided by the class of acyclic DFA. They can serve as monitors for the weights accumulated along path fragments whose traces belong to a finite set of words, e.g., formalizing finitely many request-response patterns between processes. Obviously, the full class of DFA is even more expressive and can express conditions on the total weight accumulated along any regular pattern. As a special case we also consider DFA formalizing reachability conditions. These can be used to reason about the cost accumulated until a certain event occurs. The resulting logic is indeed very expressive and covers the core features of several other logics that have been studied in the literature [8, 26]. For a detailed discussion on related formalisms we refer to Section 3.4.

Contribution. Besides the presentation of the syntax and semantics of linear temporal logic with weight assertions, our main contribution is to study the impact of different types of weight assertions and classes of DFA on the decidability of the PMC problem and to provide sharp complexity bounds for decidable fragments.

For the class of acyclic (and window) DFA we show in Section 4 that the PMC problem for WMDPs is solvable using a reduction to the standard PMC problem for unweighted MDPs. The reduction is, however, exponential in the size of the acyclic DFA occurring in the given formula. We then study the complexity of several simple patterns of formulas with weight assertions and establish NP- or coNP-hardness for the model-checking problem in WTSs and WMCs.

The border between decidability and undecidability will be addressed in Section 5. For the full class of DFA, we immediately get undecidability of the model-checking problem by the undecidability results presented in [8] for temporal logic with assertions on the weights accumulated along prefixes of infinite paths. We strengthen this result by proving that the model-checking problem is even undecidable for propositional logic with weight assertions obtained by DFA imposing reachability conditions. This is surprising given that [8] proves decidability for Boolean combinations of the CTL-modality $\exists\Diamond$ and prefix-accumulation assertions. Furthermore, we discuss the impact of weight functions versus non-negative reward functions, multiple versus single weight functions and the type of weight constraints. For omitted proofs and further technical details see [5].

2. Preliminaries

Throughout the paper we assume the reader is familiar with temporal logics, automata over finite and infinite words and Markovian models. We provide a brief summary of the relevant concepts for Markov decision processes. Further details can be found, e.g., in [3, 15, 17, 24].

Markov decision processes (MDPs). An MDP is a tuple $\mathcal{M} = (S, Act, P, \mathsf{AP}, \mathsf{L})$, where S is a finite set of states, Act a finite set of actions, $P : S \times Act \times S \to [0,1]$, AP is a finite set of atomic propositions and $\mathsf{L} : S \to 2^{\mathsf{AP}}$ a labeling function. We require that the values $P(s, act, s')$ are rational and $\sum_{s' \in S} P(s, act, s') \in \{0,1\}$ for all states $s \in S$ and actions $act \in Act$. The triples (s, act, s') with $P(s, act, s') > 0$ are called *steps*. Action act is said to be *enabled* in state s if $P(s, act, s') > 0$ for some state s'. $Act(s)$ denotes the set of actions that are enabled in $s \in S$. To avoid terminal behaviors, we require that $Act(s) \neq \varnothing$ for all states s.

A *pointed* MDP is an MDP with a distinguished initial state s_{init}. Intuitively, the computation starts in $s_0 = s_{init}$. If after i steps the current state is s_i then $\mathcal{M}$ selects nondeterministically one of the enabled actions $act_i \in Act(s_i)$, followed by an internal probabilistic choice to move to one of the states s_{i+1} where $P(s_i, act_i, s_{i+1})$ is positive.

Paths in an MDP $\mathcal{M}$ can be seen as sample runs that are obtained in this way. Formally, the paths are finite or infinite sequences where states and actions alternate:

$$\zeta \quad = \quad s_0\, act_0\, s_1\, act_1 \ldots \in (S \times Act)^* S \cup (S \times Act)^\omega$$

with $act_i \in Act(s_i)$ and $P(s_i, act_i, s_{i+1}) > 0$ for all i. The trace of ζ is obtained by ignoring the actions and taking the projection to the state labels. If $0 \leqslant h \leqslant k$ then $\zeta[h \ldots k]$ denotes the path fragment starting in the $(h{+}1)$-st state and ending in the $(k{+}1)$-st state. Thus, if ζ is as above then:

$$\begin{aligned} trace(\zeta) \quad &= \quad \mathsf{L}(s_0)\, \mathsf{L}(s_1)\, \mathsf{L}(s_2)\ \ldots \in \big(2^{\mathsf{AP}}\big)^* \cup \big(2^{\mathsf{AP}}\big)^\omega \\ \zeta[h \ldots k] \quad &= \quad s_h\, act_h\, s_{h+1}\, act_{h+1} \ldots act_{k-1}\, s_k \end{aligned}$$

In particular, $\zeta[k]$ is the $(k{+}1)$-st state in ζ. We write $\mathit{first}(\zeta)$ to denote the first state of ζ. If π is a finite path, then $last(\pi)$ denotes its last state and $|\pi|$, the length of π, stands for the number of steps that are taken in π. $IPaths$ and $FPaths$ stand for the set of all infinite resp. finite paths.

Schedulers and induced probability space. Reasoning about probabilities for path properties in MDPs requires the selection of an initial state and the resolution of the nondeterministic choices between the possible transitions. The latter is formalized via *schedulers*, also called policies or adversaries, which take as input a finite path and select an action to be executed. A (deterministic) scheduler is a function $\mathfrak{S} : FPaths \to Act$ such that $\mathfrak{S}(\pi) \in Act\big(last(\pi)\big)$ for all finite paths π. Given an initial state s, the behavior of $\mathcal{M}$ under $\mathfrak{S}$ is purely probabilistic. Standard concepts of measure and probability theory can be applied to define a sigma-algebra and a probability measure $\Pr^{\mathfrak{S}}_{\mathcal{M},s}$ for measurable sets of the infinite paths, also called *(path) events* or *path properties*. For further details, we refer to standard text books such as [24].

Weighted MDPs (WMDP). A weight function for $\mathcal{M}$ is a function $wgt : S \times Act \to \mathbb{Q}$. We extend wgt to a function that assigns to each finite path its accumulated weight.

$$wgt(s_0\, act_0\, s_1\, act_2 \ldots act_{n-1}\, s_n) = \sum_{j=0}^{n-1} wgt(s_j, act_j)$$

The logic introduced in Section 3 will be interpreted over *weighted* MDPs (WMDPs), i.e., tuples $(S, Act, P, \mathsf{AP}, \mathsf{L}, \overline{wgt})$ consisting of an MDP and a d-tuple $\overline{wgt} = (wgt_1, \ldots, wgt_d)$ of weight functions. Non-negative weight functions $wgt : S \times Act \to \mathbb{Q}_{\geqslant 0}$ are called *reward functions*. wgt is called *positive* if $wgt(s, act) > 0$ for all $s \in S$ and $act \in Act(s)$.

Weighted Markov chains (WMC). Markov chains (MC) can be seen as special instances of MDPs where the action set is a singleton. Thus, the behavior of MCs is purely probabilistic. The action set will be dropped when talking about Markov chains. We write MCs as tuples (S, P, AP, L) and $P(s, s')$ for the transition probabilities. Paths in Markov chains are just state sequences and weight functions are functions of the type $wgt : S \to \mathbb{Q}$. Intuitively, $wgt(s)$ stands for the costs resp. the reward earned when leaving s. Thus, the weight of finite paths is given by $wgt(s_0\, s_1 \ldots s_n) = wgt(s_0) + wgt(s_1) + \ldots + wgt(s_{n-1})$. The concept of schedulers is irrelevant for MCs and we write $\Pr_{\mathcal{M},s}$ for the probability measure induced by $\mathcal{M}$ when s is viewed as the initial state.

Weighted transition systems (WTS). A WMDP where the values of the transition probabilities are 0 or 1 can be seen as a weighted transition system, briefly called WTS. We write $s \xrightarrow{act} s'$ if $P(s, act, s') = 1$.

With abuse of notations, the abbreviations WMDP, WMC and WTS are often used for pointed structures.

Deterministic finite automata (DFA). The logic introduced in the next section will use DFA serving as monitors for the accumulated weights in a WMDP. In this context, each DFA is given by a tuple $\mathcal{A} = (Q, \delta, q_{init}, F)$ where Q is a finite set of states, $\delta : Q \times 2^{\mathsf{AP}} \to Q$ a partial transition function, $q_{init} \in Q$ the initial state and $F \subseteq Q$ a set of final states. We write $\mathcal{L}(\mathcal{A})$ for the accepted language.

If ϕ is a propositional formula over AP then $\mathcal{A}[\ldots\phi]$ denotes the minimal DFA where $\mathcal{L}(\mathcal{A}[\ldots\phi])$ consists of all finite words $\mathsf{A}_1\,\mathsf{A}_2 \ldots \mathsf{A}_n$ over 2^{AP} with $\mathsf{A}_n \models \phi$. Similarly, $\mathcal{A}[\phi\ldots]$ and $\mathcal{A}[\phi_1 \ldots \phi_2]$ denote minimal DFA accepting precisely the words $\mathsf{A}_1\,\mathsf{A}_2 \ldots \mathsf{A}_n$ with $\mathsf{A}_1 \models \phi$ and the words $\mathsf{A}_1\,\mathsf{A}_2 \ldots \mathsf{A}_n$ with $\mathsf{A}_1 \models \phi_1$ and $\mathsf{A}_n \models \phi_2$.

For $\ell \in \mathbb{N}$, $\ell \geqslant 1$, let $\mathcal{A}_{\leqslant \ell}$ and $\mathcal{A}_{=\ell}$ denote minimal DFA for the languages consisting of all words over the alphabet 2^{AP} of length at most $\ell{+}1$ resp. of length precisely $\ell{+}1$.

3. LTL with monitored weight assertions

Linear temporal logic with monitored weight assertions extends standard LTL by two new modalities $\diamondminus$ and $\diamondplus$ that impose linear constraints on the accumulated weights along finite paths that meet a regular condition given by a DFA.

3.1 Syntax

A signature Sig for the linear temporal logic with monitored weight constraints consists of finitely many weight symbols $\mathsf{wgt}_1, \ldots, \mathsf{wgt}_d$, a finite set AP of atomic propositions and a class AUT consisting of DFA over the alphabet 2^{AP}. We consider here the following classes AUT:

Window:	the class of DFA $\mathcal{A}_{=\ell}$ and $\mathcal{A}_{\leqslant \ell}$ for $\ell \geqslant 1$
Acyc:	the class of acyclic DFA
Reach:	the class of DFA of the form $\mathcal{A}[\ldots\phi]$
All:	the full class of DFA

For all classes, we require additionally that all DFA $\mathcal{A} \in \mathsf{AUT}$ are minimal and that the accepted language $\mathcal{L}(\mathcal{A})$ is nonempty and does not contain the empty word ε. The assumption that $\mathcal{A}$ is minimal and $\mathcal{L}(\mathcal{A})$ is nonempty implies that all states $q \in Q$ can reach F. The requirement that $\mathcal{A}$ does not accept the empty word ε is equivalent to the requirement that $q_{init} \notin F$. For each DFA $\mathcal{A} \in \mathsf{Acyc}$ its language $\mathcal{L}(\mathcal{A})$ is finite and the length of the longest run is the length of a longest word in $\mathcal{L}(\mathcal{A})$. The accepted language of each DFA $\mathcal{A}[\ldots\phi] \in \mathsf{Reach}$ can be seen as a reachability condition $\Diamond\phi$.

A *basic weight constraint* over Sig is a constraint of the form $\mathsf{expr} \bowtie c$. Here, $\bowtie$ is one of the four comparison operators $<$, $>$, $\leqslant$ or $\geqslant$, $c \in \mathbb{Q}$ and expr is a *weight expression* of the form

$$\mathsf{expr} \;=\; \sum_{i=1}^{d} a_i \cdot \mathsf{wgt}_i \quad \text{with coefficients } a_i \in \mathbb{Q}.$$

A *weight constraint* is a Boolean combination of basic weight constraints. The class WC of weight constraints is closed under negation, and so is the class of basic weight constraints as, e.g., $\neg(\mathsf{expr} \leqslant c)$ is equivalent to $\mathsf{expr} > c$.

A weight expression is called *simple* if it has the form wgt_i for some $i \in \{1, \ldots d\}$. Simple basic weight constraints have the form $\mathsf{wgt}_i \bowtie c$. A weight constraint constr is said to be simple if all its weight expressions are simple. Obviously, for $d{=}1$ all weight constraints are equivalent to simple ones.

The logic $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$ extends LTL by two new modalities $\diamondplus$ and $\diamondminus$ to formalize constraints on the accumulated weight of path fragments. For the LTL fragment we use standard temporal modalities U (until) and S (since). The previous and next operators are omitted from the basis syntax since they are derivable (see Remark 2). The abstract syntax of $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$-formulas is defined as follows:

$$\begin{aligned} \varphi \quad ::= \quad & \mathsf{tt} \;\mid\; \mathsf{a} \;\mid\; \neg\varphi \;\mid\; \varphi_1 \wedge \varphi_2 \;\mid\; \varphi_1 \,\mathrm{S}\, \varphi_2 \;\mid\; \varphi_1 \,\mathrm{U}\, \varphi_2 \;\mid \\ & \diamondminus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2) \;\mid\; \diamondplus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2) \end{aligned}$$

where $\mathsf{a} \in \mathsf{AP}$, $\mathcal{A} \in \mathsf{AUT}$, $\mathsf{constr} \in \mathsf{WC}$ and φ_1 and φ_2 are again $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$-formulas. We refer to $\diamondminus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ as *(generalized) weight assertion*. We simply write $\diamondminus^{\mathcal{A}}\,\mathsf{constr}$ for $\diamondminus^{\mathcal{A}}(\mathsf{tt}; \mathsf{constr}; \mathsf{tt})$ and refer to formulas of this type as *pure weight assertions*. Formulas of the form $\diamondminus^{\mathcal{A}}(\mathsf{expr} \bowtie c)$, are called *basic weight assertions*. Generalized, pure and basic weight assertions using the $\diamondplus$-modality are defined accordingly.

While $\diamondminus^{\mathcal{A}}$ is a past operator, $\diamondplus^{\mathcal{A}}$ imposes a constraint on the future behavior. Intuitively, $\diamondminus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ asserts that for the current position k of a path ζ, there is a path fragment $\pi = \zeta[h \ldots k]$ ending in the current position accepted by $\mathcal{A}$ such that π satisfies constr and the precondition φ_1 holds in the h-th position of ζ as well as the postcondition φ_2 in its k-th position. Similarly, $\diamondplus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ imposes a constraint on the weights that will be accumulated along some path fragment that is accepted by $\mathcal{A}$, satisfying the precondition φ_1 and the postcondition φ_2 in the position of their first and last state respectively. Thus, $\mathcal{A}$ can be viewed as a monitor that observes the traces of $\mathcal{M}$. For example, $\mathcal{A}$ can be used to formalize requirements on the accumulated weights between sending a request and receiving a response.

Length of formulas. The length of an $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$-formula φ is defined as the number of occurrences of logical operators $\neg$, $\wedge$, S and U plus the length of all generalized weight assertions that appear in φ. The length of $\diamondminus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ or $\diamondplus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ is the number of states in $\mathcal{A}$ plus the sum of the lengths of the precondition φ_1, the postcondition φ_2 and the constraint constr. The latter is defined as the sum of the lengths of the binary encodings of the coefficients $a_1, \ldots, a_d$ and the constant c in the basic weight constraints $(a_1 \cdot \mathsf{wgt}_1 + \ldots + a_d \cdot \mathsf{wgt}_d) \bowtie c$ of constr.

Derived operators. As usual we can derive all operators from propositional logic (disjunction $\vee$, implication $\to$, etc.). The temporal modalities $\Diamond$ (eventually), $\Box$ (always) and R (release) can be derived as in standard LTL by $\Diamond\varphi \overset{\text{def}}{=} \mathsf{tt}\,\mathrm{U}\,\varphi$, $\Box\varphi \overset{\text{def}}{=} \neg\Diamond\neg\varphi$ and $\varphi_1 \,\mathrm{R}\, \varphi_2 \overset{\text{def}}{=} \neg(\neg\varphi_1 \,\mathrm{U}\, \neg\varphi_2)$.

$(\zeta, k) \models \mathsf{a}$	iff	$\mathsf{a} \in \mathsf{L}\big(\zeta[k]\big)$	$(\zeta, k) \models \mathsf{tt}$
$(\zeta, k) \models \neg\varphi$	iff	$(\zeta, k) \not\models \varphi$	
$(\zeta, k) \models \varphi_1 \wedge \varphi_2$	iff	$(\zeta, k) \models \varphi_1$ and $(\zeta, k) \models \varphi_2$	
$(\zeta, k) \models \varphi_1 \,\mathrm{U}\, \varphi_2$	iff	there exists $h \geqslant k$ such that $(\zeta, h) \models \varphi_2$ and $(\zeta, i) \models \varphi_1$ for $k \leqslant i < h$	
$(\zeta, k) \models \varphi_1 \,\mathrm{S}\, \varphi_2$	iff	there exists $h \leqslant k$ such that $(\zeta, h) \models \varphi_2$ and $(\zeta, i) \models \varphi_1$ for $k \geqslant i > h$	
$(\zeta, k) \models \diamondminus^{\mathcal{A}} (\varphi_1; \mathsf{constr}; \varphi_2)$	iff	there exists $h \leqslant k$ s.t. $\mathit{trace}(\zeta[h \ldots k]) \in \mathcal{L}(\mathcal{A})$, $\zeta[h \ldots k] \models \mathsf{constr}$ and $(\zeta, h) \models \varphi_1$ and $(\zeta, k) \models \varphi_2$	
$(\zeta, k) \models \diamondplus^{\mathcal{A}} (\varphi_1; \mathsf{constr}; \varphi_2)$	iff	there exists $h \geqslant k$ s.t. $\mathit{trace}(\zeta[k \ldots h]) \in \mathcal{L}(\mathcal{A})$, $\zeta[k \ldots h] \models \mathsf{constr}$ and $(\zeta, k) \models \varphi_1$ and $(\zeta, h) \models \varphi_2$	

Figure 1. Semantics of $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$ over infinite paths ζ and position $k \in \mathbb{N}$

We can also ask whether all path fragments accepted by a monitor automaton fulfill some weight constraint instead of just one. The formula

$$\boxminus^{\mathcal{A}} \mathsf{constr} \stackrel{\text{def}}{=} \neg \diamondminus^{\mathcal{A}} \neg \mathsf{constr}$$

states that each suffix of the current system history accepted by $\mathcal{A}$ fulfills constr. Similarly,

$$\boxplus^{\mathcal{A}} \mathsf{constr} \stackrel{\text{def}}{=} \neg \diamondplus^{\mathcal{A}} \neg \mathsf{constr}$$

asserts that each future behavior accepted by $\mathcal{A}$ satisfies constr.

Sublogics. We write $\mathrm{PL}[\diamondplus : \mathsf{AUT}]$ for propositional logic where the atoms are future pure weight assertions, i.e., formulas of $\mathrm{PL}[\diamondplus : \mathsf{AUT}]$ are Boolean combinations of formulas of the type $\diamondplus^{\mathcal{A}} \mathsf{constr}$. Note that the analogous logic $\mathrm{PL}[\diamondminus : \mathsf{AUT}]$ is pointless as it is as expressive as propositional logic. $\mathrm{LTL}_{\mathrm{simple}}[\diamondplus, \diamondminus : \mathsf{AUT}]$ denotes $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$ restricted to simple weight constraints.

3.2 Semantics

Formulas of the logic $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$ can be interpreted over structures consisting of directed graphs with a d-dimensional weight function and node-labels in AP. Here, we deal with an MDP-semantic of $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$ and interpret formulas over the infinite paths of a WMDP $\mathcal{M} = (S, \mathit{Act}, P, \mathsf{AP}, \mathsf{L}, \overline{\mathit{wgt}})$ with $\overline{\mathit{wgt}} = (\mathit{wgt}_1, \ldots, \mathit{wgt}_d)$ as in Section 2. Weight expressions are evaluated over the finite paths in $\mathcal{M}$ in the expected way. Given a basic weight constraint $\mathsf{expr} \bowtie c$ and a finite path π in $\mathcal{M}$ we define:

$$\pi \models \mathsf{expr} \bowtie c \quad \text{iff} \quad [\![\mathsf{expr}, \pi]\!] \bowtie c$$

where $[\![\mathsf{expr}, \pi]\!]$ denotes the value of the weight expression expr when interpreting the weight symbols wgt_i with the accumulated weight of π under weight function wgt_i:

$$[\![\mathsf{expr}, \pi]\!] \stackrel{\text{def}}{=} \sum_{i=1}^{d} a_i \cdot \mathit{wgt}_i(\pi) \text{ for } \mathsf{expr} = \sum_{i=1}^{d} a_i \cdot \mathsf{wgt}_i$$

The satisfaction relation $\models$ for finite paths and weight constraints (i.e., Boolean combination of basic weight constraints) is now defined in the obvious way. The interpretation of $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$-formulas in the WMDP $\mathcal{M}$ is defined over pairs (ζ, k) where $\zeta = s_0\, \mathit{act}_0\, s_1\, \mathit{act}_1\, s_2\, \mathit{act}_2 \ldots$ is an infinite path in $\mathcal{M}$ and $k \in \mathbb{N}$ as shown in Figure 1. Thus, $(\zeta, k) \models \diamondminus^{\mathcal{A}} \mathsf{constr}$ iff there exists $h \leqslant k$ with $\mathit{trace}(\zeta[h \ldots k]) \in \mathcal{L}(\mathcal{A})$ and $\zeta[h \ldots k] \models \mathsf{constr}$. Similarly, $(\zeta, k) \models \boxminus^{\mathcal{A}} \mathsf{constr}$ iff for each $h \leqslant k$ we have: $\mathit{trace}(\zeta[h \ldots k]) \notin \mathcal{L}(\mathcal{A})$ or $\zeta[h \ldots k] \models \mathsf{constr}$. The semantics of future pure weight assertions is analogous. To reason about the probabilities for properties specified in $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$ we lift the semantics to infinite paths:

$$\zeta \models \varphi \quad \text{iff} \quad (\zeta, 0) \models \varphi$$

Remark 1 (Distributivity) Weight constraints can be arbitrary Boolean combinations of basic weight constraints. For disjunctive weight constraints we can rely on the distributivity law for *existential* quantification and disjunction and obtain:

$$\begin{aligned} & \diamondminus^{\mathcal{A}} \big(\varphi_1; \mathsf{constr}_1 \vee \mathsf{constr}_2; \varphi_2 \big) \\ \equiv \quad & \diamondminus^{\mathcal{A}} (\varphi_1; \mathsf{constr}_1; \varphi_2) \ \vee\ \diamondminus^{\mathcal{A}} (\varphi_1; \mathsf{constr}_2; \varphi_2) \end{aligned}$$

where $\equiv$ denotes the equivalence of formulas. By duality, we get the equivalence of the formulas $\boxminus^{\mathcal{A}} \mathsf{constr}_1 \wedge \mathsf{constr}_2$ and $\boxminus^{\mathcal{A}} \mathsf{constr}_1 \wedge \boxminus^{\mathcal{A}} \mathsf{constr}_2$. The analogous statements hold for $\diamondplus$ and $\boxplus$. ■

Remark 2 (Pre-/postconditions, past vs. future) The postcondition in past weight assertions and the precondition in future weight assertions impose a constraint on the current position. Hence:

$$\begin{aligned} \diamondminus^{\mathcal{A}} (\varphi_1; \mathsf{constr}; \varphi_2) \quad &\equiv \quad \diamondminus^{\mathcal{A}} (\varphi_1; \mathsf{constr}; \mathsf{tt}) \wedge \varphi_2 \\ \diamondplus^{\mathcal{A}} (\varphi_1; \mathsf{constr}; \varphi_2) \quad &\equiv \quad \diamondplus^{\mathcal{A}} (\mathsf{tt}; \mathsf{constr}; \varphi_2) \wedge \varphi_1 \end{aligned}$$

In combination with an eventually operator, the semantics of $\diamondminus$ and $\diamondplus$ coincide. That is,

$$\Diamond \diamondminus^{\mathcal{A}} (\varphi_1; \mathsf{constr}; \varphi_2) \equiv \Diamond \diamondplus^{\mathcal{A}} (\varphi_1; \mathsf{constr}; \varphi_2).$$

Note that for each infinite path ζ:

$$\begin{aligned} & \zeta \models \Diamond \diamondminus^{\mathcal{A}} (\varphi_1; \mathsf{constr}; \varphi_2) \\ \text{iff} \quad & \zeta \models \Diamond \diamondplus^{\mathcal{A}} (\varphi_1; \mathsf{constr}; \varphi_2) \\ \text{iff} \quad & \text{there exists } h, k \in \mathbb{N} \text{ with } h \leqslant k \text{ such that} \\ & \quad (1) \quad \mathit{trace}(\zeta[h \ldots k]) \in \mathcal{L}(\mathcal{A}) \\ & \quad (2) \quad \zeta[h \ldots k] \models \mathsf{constr} \\ & \quad (3) \quad (\zeta, h) \models \varphi_1 \\ & \quad (4) \quad (\zeta, k) \models \varphi_2 \end{aligned}$$

By duality, $\Box \boxminus^{\mathcal{A}} \mathsf{constr} \ \equiv\ \Box \boxplus^{\mathcal{A}} \mathsf{constr}$. ■

Remark 3 (Window weight assertions, next and previous) Step-bounded properties can be expressed using the automata class Window. The DFA $\mathcal{A}_{\leqslant \ell}$ and $\mathcal{A}_{=\ell}$ can be seen as monitors that observe paths up to length ℓ or of length precisely ℓ. In what follows, $\diamondminus^{\leqslant \ell}$ and $\diamondminus^{=\ell}$ are used as brief notations for $\diamondminus^{\mathcal{A}_{\leqslant \ell}}$ and $\diamondminus^{\mathcal{A}_{=\ell}}$ respectively, and called (past) *window weight operators*. Future window weight operators are defined accordingly. The standard next and previous operators are definable using generalized window weight assertions:

$$\bigcirc \varphi \quad \stackrel{\text{def}}{=} \quad \diamondplus^{=1}(\mathsf{tt}; \mathrm{true}; \varphi)$$

The previous operator $\ominus \varphi$ is obtained by $\diamondminus^{=1}(\varphi; \mathrm{true}; \mathsf{tt})$. ■

Remark 4 (Existential vs. universal window weight assertions) Existential and universal pure future window weight assertions agree for precise window length, i.e., $\diamondplus^{=\ell}\,\mathsf{constr} \;\equiv\; \boxplus^{=\ell}\mathsf{constr}$, as we have for all path-position pairs (ζ, k):

$$
\begin{array}{rll}
(\zeta,k) \models \diamondplus^{=\ell}\,\mathsf{constr} & \text{iff} & (\zeta,k) \models \boxplus^{=\ell}\mathsf{constr} \\
& \text{iff} & \zeta[k \ldots k{+}\ell] \models \mathsf{constr}
\end{array}
$$

In contrast, the formulas $\diamondminus^{=\ell}\,\mathsf{constr}$ and $\boxminus^{=\ell}\mathsf{constr}$ are not equivalent since for $k < \ell$ and each infinite path ζ we have $(\zeta,k) \models \boxminus^{=\ell}\mathsf{constr}$, while $(\zeta,k) \not\models \diamondminus^{=\ell}\,\mathsf{constr}$.

However, if $k \geqslant \ell$ then for each infinite path ζ:

$$
\begin{array}{rll}
(\zeta,k) \models \diamondminus^{=\ell}\,\mathsf{constr} & \text{iff} & (\zeta,k) \models \boxminus^{=\ell}\mathsf{constr} \\
& \text{iff} & \zeta[k{-}\ell \ldots k] \models \mathsf{constr}
\end{array}
$$

In combination with prefix-independent temporal modalities the effect of $\boxminus$ and $\diamondminus$ collapses, e.g.:

$$
\begin{array}{rcl}
\Box\Diamond\,\diamondminus^{=\ell}\,\mathsf{constr} & \equiv & \Box\Diamond\,\boxminus^{=\ell}\,\mathsf{constr} \\
\Diamond\Box\,\diamondminus^{=\ell}\,\mathsf{constr} & \equiv & \Diamond\Box\,\boxminus^{=\ell}\,\mathsf{constr}
\end{array}
$$

■

Remark 5 (Step- and weight-bounded until and release) Interpreted over a structure with a single weight function wgt, the formula $\mathsf{a}\,\mathrm{U}^{\leqslant\ell}_{\bowtie c}\,\mathsf{b}$ is a variant of $\mathsf{a}\,\mathrm{U}\,\mathsf{b}$ with step bound ℓ and weight constraint $\mathsf{wgt} \bowtie c$. Formally, if $\ell \geqslant 1$ then $(\zeta,k) \models \mathsf{a}\,\mathrm{U}^{\leqslant\ell}_{\bowtie c}\,\mathsf{b}$ iff there exists $k \leqslant h \leqslant k+l$ with $wgt(\zeta[k \ldots h]) \bowtie c$ such that $\mathsf{b} \in \mathsf{L}(\zeta[h])$ and $\mathsf{a} \in \mathsf{L}(\zeta[i])$ for $k \leqslant i < h$. Let $\mathcal{A}[\mathsf{a}\,\mathrm{U}^{\leqslant\ell}\,\mathsf{b}] \in \mathsf{Acyc}$ be a DFA for the language consisting of the words $\mathsf{A}_1\,\mathsf{A}_2 \ldots \mathsf{A}_n$ over 2^{AP} such that $n \leqslant \ell{+}1$, $\mathsf{a} \in \mathsf{A}_i$ for $0 \leqslant i < n$ and $\mathsf{b} \in \mathsf{A}_n$. Then:

$$
\mathsf{a}\,\mathrm{U}^{\leqslant\ell}_{\bowtie c}\,\mathsf{b} \quad \equiv \quad \diamondplus^{\mathcal{A}[\mathsf{a}\,\mathrm{U}^{\leqslant\ell}\,\mathsf{b}]}(\mathsf{wgt} \bowtie c)
$$

and by duality we get:

$$
\mathsf{a}\,\mathrm{R}^{\leqslant\ell}_{\bowtie c}\,\mathsf{b} \quad \equiv \quad \boxplus^{\mathcal{A}[\neg\mathsf{a}\,\mathrm{U}^{\leqslant\ell}\,\neg\mathsf{b}]}\neg(\mathsf{wgt} \bowtie c).
$$

■

Interpretation over WMDP, WMC, WTS. Our main interest is in reasoning about the probabilities of $\mathrm{LTL}[\diamondplus,\diamondminus : \mathsf{AUT}]$-specifications φ in weighted Markovian models. If $(\mathcal{M}, s)$ is a WMC then:

$$
\Pr_{\mathcal{M},s}(\varphi) \;\stackrel{\text{def}}{=}\; \Pr_{\mathcal{M},s}\big\{\, \zeta \in \mathit{IPaths} \,:\, \zeta \models \varphi \,\big\}
$$

Similarly, if $(\mathcal{M}, s)$ is a WMDP and $\mathfrak{S}$ a scheduler for $\mathcal{M}$ then $\Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi)$ denotes the probability for φ under scheduler $\mathfrak{S}$ and for starting state s. As usual, we define:

$$
\Pr^{\max}_{\mathcal{M},s}(\varphi) \stackrel{\text{def}}{=} \sup_{\mathfrak{S}} \Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi), \quad \Pr^{\min}_{\mathcal{M},s}(\varphi) \stackrel{\text{def}}{=} \inf_{\mathfrak{S}} \Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi)
$$

where $\mathfrak{S}$ ranges over all schedulers for $\mathcal{M}$. When interpreting $\mathrm{LTL}[\diamondplus,\diamondminus : \mathsf{AUT}]$-formulas over WTS, we use CTL-like notations such as $s \models \exists\varphi$ to indicate the existence of an infinite path ζ starting in state s with $\zeta \models \varphi$.

Notation 6 (Model-checking problem) To discuss the complexity and decidability we will study decision variants of the model-checking problem for $\mathrm{LTL}[\diamondplus,\diamondminus : \mathsf{AUT}]$ interpreted over WMDP, WMC or WTS. For WMDP the notion model-checking problem will be used to refer to one of the four problems asking whether (a), (b), (c) or (d) holds, where

$$
\begin{array}{llll}
\text{(a)} & \Pr^{\max}_{\mathcal{M},s_{init}}(\varphi) > 0 & \text{(c)} & \Pr^{\min}_{\mathcal{M},s_{init}}(\varphi) > 0 \\
\text{(b)} & \Pr^{\max}_{\mathcal{M},s_{init}}(\varphi) = 1 & \text{(d)} & \Pr^{\min}_{\mathcal{M},s_{init}}(\varphi) = 1
\end{array}
$$

(a) and (d) are trivially interreducible since

$$
\Pr^{\min}_{\mathcal{M},s_{init}}(\varphi) = 1 - \Pr^{\max}_{\mathcal{M},s_{init}}(\neg\varphi).
$$

The analogous statement holds for problems (b) and (c). We refer to (a) as the *positive* model-checking problem and to (b) as the *almost-sure* model-checking problem. For WMC all four problems collapse from a computational point of view since the concept of schedulers is irrelevant. For WTS, the task of the *existential* model-checking problem is to decide whether $s_{init} \models \exists\varphi$ for a given formula φ, while the *universal* model-checking problem asks whether $s_{init} \models \forall\varphi$. ■

Remark 7 (Integer vs. rational weights) We introduced weight functions in MDPs as functions of the type $wgt : S \times Act \to \mathbb{Q}$. When using $\mathrm{LTL}[\diamondplus,\diamondminus : \mathsf{AUT}]$ as a specification formalism for finite WMDPs, however, integer-valued weight functions are equally expressive. ■

Remark 8 (Transformations of weight functions) Using the idea of [8], each basic weight constraints $\mathsf{expr} \bowtie c$ where $\mathsf{expr} = a_1 \cdot \mathsf{wgt}_1 + \ldots + a_d \cdot \mathsf{wgt}_d$ can be replaced with a simple basic weight constraint $\mathsf{wgt} \bowtie c$ where wgt is a fresh weight symbol representing a new weight function $wgt : S \times Act \to \mathbb{Q}$ where $wgt(s, act) = \sum_{i=1}^{d} a_i wgt(s, act)$. ■

3.3 Examples

To illustrate the expressiveness and usefulness of our formalism we provide a number of examples from different domains. In what follows, we use some shorthand notations for weight constraints with obvious meanings. For instance, $\mathsf{wgt} = c$ is short for $(\mathsf{wgt} \leqslant c) \wedge (\mathsf{wgt} \geqslant c)$ and $c_1 \leqslant \mathsf{wgt} \leqslant c_2$ means $(\mathsf{wgt} \geqslant c_1) \wedge (\mathsf{wgt} \leqslant c_2)$.

The following formula specifies global bounds on the load, measured in the number of incoming requests to a system within a given monitor $\mathcal{A}$, is between c_{min} and c_{max}.

$$
\Box\, \boxminus^{\mathcal{A}[\mathsf{begin}\ldots\mathsf{end}]}\, (c_{\min} \leqslant \mathsf{load} \leqslant c_{\max})
$$

The property that whenever there is a request to the system, the utility of the system exceeds a certain threshold c within monitor $\mathcal{A}$ is asserted by the following formula.

$$
\Box\big(\, \mathsf{request} \;\to\; \boxplus^{\mathcal{A}}(\mathsf{utility} \geqslant c)\,\big)
$$

Similarly, the formula

$$
\Box(\, \mathsf{request} \to \diamondplus^{\leqslant\ell}(\mathsf{utility} \geqslant c)\,)
$$

can be used to specify that after each request a utility value of at least c is guaranteed within ℓ or fewer steps.

The following example formalizes a resilience property, in which we require that whenever in the last ten rounds of some protocol, the accumulated number of errors a certain component produced exceeded five, the component will receive no load until the replacement procedure (formalized by $\mathcal{A}$) is complete.

$$
\diamondminus^{=10}(\mathrm{error} > 5) \to \boxplus^{\mathcal{A}}(\mathrm{load} = 0)
$$

Using two (or more) different weight functions allows, e.g., to reason about the load balancing of two (or more) subsystems.

$$
\Box\big(\, \diamondminus^{\mathcal{A}}(\, |\mathsf{load}_1 - \mathsf{load}_2| \leqslant c\,)\,\big)
$$

The formula above states that globally, within a given monitor $\mathcal{A}$ the load difference must not exceed a certain threshold c.

With two weight functions one can also express properties related to the tradeoff between cost and utility. E.g., the following formula might state that whenever the system consumes a certain amount of energy c_e for processing a query then the accumulated utility exceeds some utility threshold c_u:

$$
\Box\big(\, \diamondminus^{\mathcal{A}}(\mathsf{energy} \geqslant c_e) \to \diamondminus^{\mathcal{A}}(\mathsf{utility} \geqslant c_u)\,\big)
$$

Nesting of formulas allows, e.g., expressing properties of the following type. For this, let $\mathcal{A}_{init}$ formalize an initialization process and $\mathcal{A}_{work}$ a working phase.

Then the formula:

$$\diamondplus^{\mathcal{A}_{init}}(\mathsf{tt}; \mathsf{energy} < c_e; \diamondplus^{\mathcal{A}_{work}}(\mathsf{utility} \geqslant c_u))$$

stands for the requirements that there is an initialization process which uses not more than c_e energy and is followed by a working phase which in turn gains at least utility c_u.

Assertions on the ratio of two weight functions can be expressed using weight expressions. For example, the following formula expresses that the monitored ratio of utility and energy exceeds some threshold c:

$$\Box\big(\boxminus^{\mathcal{A}}\big(\tfrac{\mathsf{utility}}{\mathsf{energy}} \geqslant c\big)\big)$$

where $\frac{util}{\mathsf{energy}} \geqslant c$ is a short form notation for the weight expression $\mathsf{utility} - c \cdot \mathsf{energy} \geqslant 0$.

3.4 Variants and related logics

Average. Up to now the semantics of weight expressions over finite paths is based on the accumulated weight given by the sum of all state-action pairs along the given finite path. Alternatively one might deal with the average defined by

$$\mathrm{avg}[wgt](\pi) = wgt(\pi)/|\pi|$$

if $|\pi| > 0$. Let $\mathrm{LTL}^{\mathrm{avg}}[\diamondplus, \diamondminus : \mathsf{AUT}]$ denote the extension of $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$ where basic weight constraints either have the form $\mathsf{expr} \bowtie c$ as before or $\mathsf{avgexpr} \bowtie c$ where

$$\mathsf{avgexpr} \;=\; \sum_{i=1}^{d} a_i \cdot \mathsf{avg}[\mathsf{wgt}_i] \quad \text{with } a_i \in \mathbb{Q}$$

is an *average weight expression*. The symbol $\mathsf{avg}[\mathsf{wgt}_i]$ indicates that the weight function represented by wgt_i will be interpreted by the average weight of finite paths. To ensure that the average weight of all finite paths π with $trace(\pi) \in \mathcal{L}(\mathcal{A})$ is well-defined, we impose the side constraint that for all subformulas $\diamondminus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ and $\diamondplus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$, where constr contains an average weight constraint, the DFA $\mathcal{A}$ does not accept words of length 1. Following the idea described in [8], average weight expressions can be transformed into sum weight expressions of the form $\mathsf{wgt} \bowtie 0$. This transformation is applicable for our purposes as well. This yields that the (probabilistic) model-checking problem for $\mathrm{LTL}^{\mathrm{avg}}[\diamondplus, \diamondminus : \mathsf{AUT}]$-formulas is reducible to the one for $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{AUT}]$.

Indeed several authors considered logics or specific properties that are in the spirit of or even expressible in $\mathrm{LTL}^{\mathrm{avg}}[\diamondplus, \diamondminus : \mathsf{AUT}]$ for some automata class AUT.

Fixed window properties. The (direct) fixed window properties studied in [14] for non-probabilistic weighted game structures have the form $\diamondplus^{\leqslant \ell}(\mathsf{avg}[\mathsf{wgt}] \geqslant c)$, $\Box\diamondplus^{\leqslant \ell}(\mathsf{avg}[\mathsf{wgt}] \geqslant c)$ and $\Diamond\Box\diamondplus^{=\ell}(\mathsf{avg}[\mathsf{wgt}] \geqslant c)$. Thus, they are expressible in $\mathrm{LTL}^{\mathrm{avg}}[\diamondplus, \diamondminus : \mathsf{Window}]$.

Temporal logic with prefix accumulation. The concept of prefix-accumulation assertions as introduced by Boker et al [8] for weighted Kripke structures and branching-time and linear-time temporal logics is very much in the spirit of the logic $\mathrm{LTL}^{\mathrm{avg}}[\diamondplus, \diamondminus : \mathsf{All}]$. The differences between weighted Kripke structures and WTSs in our sense are mostly of a syntactic nature. Rephrased for our notations, *prefix-accumulation assertions* as in [8] can be defined as $\mathrm{LTL}^{\mathrm{avg}}[\diamondplus, \diamondminus : \mathsf{All}]$-formulas:

$$\mathsf{assert}[\mathsf{constr}] \;\stackrel{\text{def}}{=}\; \diamondminus^{\mathcal{A}[\mathsf{init}\ldots]}\, \mathsf{constr}$$

We suppose here that init is an atomic proposition that characterizes the initial state. Thus, with this side assumption, LTL with prefix-accumulation assertions as in [8] is a sublogic of $\mathrm{LTL}^{\mathrm{avg}}[\diamondplus, \diamondminus : \mathsf{All}]$. Given a DFA $\mathcal{A} \in \mathsf{All}$ imposing a regular constraint that is not LTL-definable, we cannot expect to get an LTL formula with prefix-accumulation assertions that is equivalent to $\diamondplus^{\mathcal{A}}\, \mathsf{constr}$. However, the $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Reach}]$-formula $\diamondplus^{\mathcal{A}[\ldots\phi]}\, \mathsf{constr}$ is equivalent to the formula $\Diamond(\phi \wedge \mathsf{assert}[\mathsf{constr}])$. Thus, e.g., $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Reach}]$ can be seen as a sublogic of LTL with prefix-accumulation assertions. [8] also considers a variant of prefix accumulation "controlled" by some regular expression. This approach, however, departs from the regular conditions imposed by the DFA $\mathcal{A}$ in generalized or pure weight assertions. The purpose of regular conditions in controlled prefix accumulation as in [8] relies on an alternative definition of the weight of finite paths (where the weights of certain transitions can be ignored), while the operators $\diamondminus^{\mathcal{A}}$ and $\diamondplus^{\mathcal{A}}$ impose conditions on the standard weight of finite paths satisfying a given regular constraint.

Mean-payoff, long-run averages. Several authors studied game structures or logics with mean-payoff objectives. The latter are typically defined as requirements on the limit superior or limit inferior of the accumulated weight along the prefixes of a given infinite path. Such requirements can be formalized in $\mathrm{LTL}^{\mathrm{avg}}[\diamondplus, \diamondminus : \mathsf{All}]$ by formulas of the form

$$\Diamond\Box\diamondplus^{\mathcal{A}[\mathsf{init}\ldots]}(\mathsf{avgexpr} \bowtie c) \quad \text{or} \quad \Diamond\Box\diamondplus^{\mathcal{A}[\mathsf{init}\ldots]}(\mathsf{avgexpr} \bowtie c)$$

Again, up to some minor syntactic differences, this yields an embedding of the extension of LTL with mean-payoff assertions of [8] into $\mathrm{LTL}^{\mathrm{avg}}[\diamondplus, \diamondminus : \mathsf{All}]$. [26] proposes a further extension where expressions might be polynomial and might refer to so-called characteristic properties. For the latter, our logic provides no corresponding concept. However, the switch from (linear) weight expressions to polynomial weight expressions would be possible for our framework as well. The decidability results presented for $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$ in Section 4 would not be affected as we just require that weight constraints can be evaluated efficiently over a given d-tuple of values. Since mean-payoff assertions are prefix-independent properties, the logic presented in [26] is incomparable to our logic concerning expressiveness. Neither [8] nor [26] considers probabilistic structures. WMDPs or weighted game structures with mean-payoff objectives have been considered by several authors, see, e.g., [9, 10]. Extensions of temporal logics with formulas for weight constraints in WMDPs are mostly restricted to branching-time logics such as PRCTL with reward-bounded until and release modalities (see Remark 5) or state conditions on the expected total reward [2, 17, 20]. An exception is the logic introduced in [18] with state formulas asserting that $\Pr^{\min}_{\mathcal{M}\otimes\mathcal{A}, s_{init}}(\psi \to \varphi) = 1$ where $\psi = \Box\Diamond\diamondplus^{=1}(\mathsf{wgt}_2 > 0)$, $\varphi = \Diamond\Box\diamondplus^{\mathcal{A}[\mathsf{init}\ldots]}(\mathsf{wgt}_1/\mathsf{wgt}_2 > c)$ and $\mathcal{A}$ is a DFA (so-called experiment) that runs in parallel to the WMDP $\mathcal{M}$. $\mathsf{wgt}_1, \mathsf{wgt}_2$ stand for reward functions in $\mathcal{A}$ lifted to the product $\mathcal{M} \otimes \mathcal{A}$. Intuitively, wgt_2 counts the number of successful experiments and wgt_1 the total outcome of successful experiments. This particular concept of experiments is not expressible in our logic, but inspired our work.

4. Model checking against $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$-specifications

We now address the probabilistic model-checking (PMC) problem for $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$, where we are given a $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$-formula φ, a WMDP $\mathcal{M} = (S, Act, P, \mathsf{AP}, \mathsf{L}, \overline{wgt})$ and a state $s_{init} \in S$ and the task is to compute $\Pr^{\max}_{\mathcal{M},s}(\varphi)$ or $\Pr^{\min}_{\mathcal{M},s}(\varphi)$.

We first present a general approach that relies on a reduction to the task of computing extremal probabilities for LTL formulas in (unweighted) MDPs (Section 4.1). This approach is computationally expensive and relies on a product construction. It inherently uses a refined powerset construction for the automata appearing in subformulas $\diamondminus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ or $\diamondplus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ to store the relevant information on the possible runs in $\mathcal{A}$ and the weight for the suffixes of the current history in $\mathcal{M}$. We then discuss in Section 4.2 the time complexity of the model-checking problem for $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$ and sublogics and show that no efficient algorithms can be expected that run in time polynomial in the size of the automata $\mathcal{A}$. Efficient model-checking algorithms for special patterns of LTL[$\oplus, \ominus$: Acyc]-formulas are presented in Section 4.3.

4.1 Reduction to the LTL-PMC problem

The goal is to provide a reduction from the LTL[$\oplus, \ominus$: Acyc]-PMC problem to the LTL-PMC problem. Given an LTL[$\oplus, \ominus$: Acyc]-formula φ and a WMDP $\mathcal{M} = (S, Act, P, \mathsf{AP}, \mathsf{L}, \overline{wgt})$ where $\overline{wgt} = (wgt_1, \ldots, wgt_d)$, the idea is to replace all weight assertions $\oplus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ and $\ominus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ with an until or since formula, while adding information on the possible runs in $\mathcal{A}$ for the path fragments in $\mathcal{M}$. This is done by enhancing each state s with a partial function f for each occurring automaton $\mathcal{A}$. The function tracks all the states q the automaton $\mathcal{A}$ can possibly be in after reading the trace of a path fragment ending in s, along with a vector $\bar{w}$ of the accumulated weights along this fragment.

Let $\mathcal{A}_1, \ldots, \mathcal{A}_m$ be the DFA that occur as parameters of weight assertions in φ. Recall that $\mathcal{A}_i$ are supposed to be minimal acyclic DFA over the alphabet 2^{AP}. Let ℓ_i be the number of states in a longest run in $\mathcal{A}_i$. Then, the length of each word accepted by $\mathcal{A}_i$ is at most $\ell_i{-}1$. In particular, the maximal length of a path π in $\mathcal{M}$ where $trace(\pi)$ is accepted by $\mathcal{A}_i$ is $\ell_i{-}2$. (Recall that the length $|\pi|$ of a finite path π is the number of transitions taken in π. Thus, $trace(\pi)$ consists of $|\pi|{+}1$ symbols.) We are going to construct an (unweighted) MDP

$$\widetilde{\mathcal{M}} = Monitor(\mathcal{M}, \mathcal{A}_1, \ldots, \mathcal{A}_m) = (\widetilde{S}, Act, \widetilde{P}, \widetilde{\mathsf{AP}}, \widetilde{\mathsf{L}})$$

whose states have the form $\widetilde{s} = \langle s, f_1, \ldots, f_m \rangle$ where $s \in S$ and f_i is a partial function from $\{0, 1, \ldots, \ell_i\}$ to pairs $(q, \overline{w})$ where q is a state in $\mathcal{A}_i$ and $\overline{w} \in \mathbb{Q}^d$ such that $f_i(k) = \bot$ (undefined) for at least one k. The set of all these tuples is, of course, infinite. Below we provide the definition of the state space $\widetilde{S}$ of $\widetilde{\mathcal{M}}$ which ensures that $\widetilde{\mathcal{M}}$ has only finitely many states. See Remark 9.

The actions that are enabled in state $\widetilde{s} = \langle s, f_1, \ldots, f_m \rangle$ of $\widetilde{\mathcal{M}}$ are precisely the actions in $Act(s)$. The transition probability function $\widetilde{P}$ of $\widetilde{\mathcal{M}}$ is defined as follows. Suppose that $act \in Act(s)$. Then:

$$\widetilde{P}(\langle s, f_1, \ldots, f_m \rangle, act, \langle s', f_1', \ldots, f_m' \rangle) = P(s, act, s')$$

where f_i' is the unique (act, s')-successor of $\langle s, f_i \rangle$ in $\mathcal{A}_i$ that is defined as follows. Let us now fix some $i \in \{1, \ldots, m\}$ and suppose $\mathcal{A}_i = (Q, \delta, q_{init}, F)$. Then, $f_i : \{0, 1, \ldots, \ell_i\} \to Q \times \mathbb{Q}^d$ is a partial function such that $f_i(k) = \bot$ for at least one k. The (act, s')-successor of $\langle s, f_i \rangle$ in $\mathcal{A}_i$ is the partial function $f_i' : \{0, 1, \ldots, \ell_i\} \to Q \times \mathbb{Q}^d$ where for $0 \leqslant k \leqslant \ell_i$:

- If $f_i(k) = (q, \overline{w})$ where $q \in Q$ and $q \neq q_{init}$ then $f_i'(k) = \big(\delta(q, \mathsf{L}(s')), \overline{w} + \overline{wgt}(s, act)\big)$.
- If k is the smallest index such that $f_i(k) = \bot$ then $f_i'(k) = \big(\delta(q_{init}, \mathsf{L}(s')), \overline{0}\big)$.
- In all other cases: $f_i(k) = \bot$.

Here, we identify the tuples $(\bot, \overline{w})$ with $\bot$. Furthermore, we define the initial function f_i^s by $f_i^s(0) = (\delta(q_{init}, \mathsf{L}(s)), \overline{0})$ and $f_i^s(k) = \bot$ for $k \in \{1, \ldots, \ell_i\}$.

In all other cases, $\widetilde{P}(\cdot) = 0$. The state space $\widetilde{S}$ of $\widetilde{\mathcal{M}}$ is the smallest set that contains the states $\widetilde{s} \overset{\text{def}}{=} \langle s, f_1^s, \ldots, f_m^s \rangle$ for all $s \in S$ and that is closed under the steps induced by the transition probability function $\widetilde{P}$.

Remark 9 (Size of the state space) The set $\widetilde{S}$ is indeed finite. Specifically, for $\ell = \max\{\ell_1, \ldots, \ell_m\}$ we have

$$\big|\widetilde{S}\big| < |S|^{\ell} \cdot |Act|^{\ell-1} \cdot 2^{m \cdot (\ell+1) \cdot \log(\ell+1)}.$$

This bound is obtained by the observation that $\widetilde{S}$ can be written as the union of sets $\widetilde{S}_\pi$, where π is a path fragment of length at most $\ell{-}2$ in $\mathcal{M}$ and all states in $\widetilde{S}_\pi$ have the form $\langle last(\pi), f_1, \ldots, f_m \rangle$ where $\{f_i(0), \ldots, f_i(\ell_i)\} \setminus \{\bot\}$ consists of the pairs $(\delta(q_{init}, trace(\pi')), \overline{wgt}(\pi))$ for some prefix π' of π. This yields

$$|\widetilde{S}_\pi| \leqslant (\ell{+}1)! < 2^{m \cdot (\ell+1) \cdot \log(\ell+1)}.$$

The factor $|S|^{\ell} \cdot |Act|^{\ell-1}$ is an upper bound for the number of path fragments of length $\ell{-}2$. ■

The set $\widetilde{\mathsf{AP}}$ of atomic propositions in $\widetilde{\mathcal{M}}$ consists of:

- the atomic propositions in AP that appear in φ,
- fresh symbols $\mathsf{init}_i(k)$, $\mathsf{run}_i(k)$ and $\mathsf{goal}_i(k)$ for $i = 1, \ldots, m$ and $k \in \{0, 1, \ldots, \ell_i\}$

The labeling function $\widetilde{\mathsf{L}} : \widetilde{S} \to 2^{\widetilde{\mathsf{AP}}}$ is then defined by the following conditions. Let $\widetilde{s} = \langle s, f_1, \ldots, f_m \rangle \in \widetilde{S}$. Then $\mathsf{AP} \cap \widetilde{\mathsf{L}}(\widetilde{s}) = \mathsf{AP} \cap \mathsf{L}(s)$. For $i \in \{1, \ldots, m\}$ the semantics of $\oplus^{\mathcal{A}_i}(\varphi_1; \mathsf{constr}; \varphi_2)$ or $\ominus^{\mathcal{A}_i}(\varphi_1; \mathsf{constr}; \varphi_2)$ will be encoded using the atomic propositions $\mathsf{init}_i(k)$, $\mathsf{run}_i(k)$ and $\mathsf{goal}_i(k)$. The requirements for the labeling function is as follows where we suppose that $\mathcal{A}_i = (Q, \delta, q_{init}, F)$:

$$\begin{aligned}
\mathsf{init}_i(k) \in \widetilde{\mathsf{L}}(\widetilde{s}) \quad &\text{iff} \quad f_i(k) = (\delta(q_{init}, \mathsf{L}(s)), \overline{0}) \\
\mathsf{run}_i(k) \in \widetilde{\mathsf{L}}(\widetilde{s}) \quad &\text{iff} \quad f_i(k) \neq \bot \\
\mathsf{goal}_i(k) \in \widetilde{\mathsf{L}}(\widetilde{s}) \quad &\text{iff} \quad f_i(k) = (q, \overline{w}) \text{ for some } q \in F \\
&\phantom{\text{iff}} \quad \text{and } \mathsf{constr}[\overline{\mathsf{wgt}}/\overline{w}]
\end{aligned}$$

We use $\mathsf{constr}[\overline{\mathsf{wgt}}/\overline{w}]$ to denote the variable-free arithmetic condition resulting from constr by replacing the weight symbols wgt_k in the weight expressions of constr with the values w_k for $k = 1, \ldots, d$. Thus, $\mathsf{constr}[\overline{\mathsf{wgt}}/\overline{w}]$ can be treated as a truth value.

Let $\widetilde{\varphi}$ be the LTL formula that results from φ by replacing the subformulas $\psi_+ = \oplus^{\mathcal{A}_i}(\varphi_1; \mathsf{constr}; \varphi_2)$ and $\psi_- = \ominus^{\mathcal{A}_i}(\varphi_1; \mathsf{constr}; \varphi_2)$ with:

$$\widetilde{\psi}_+ \overset{\text{def}}{=} \bigvee_{0 \leqslant k \leqslant \ell_i} \big(\varphi_1 \wedge \mathsf{init}_i(k) \wedge (\mathsf{run}_i(k) \,\mathrm{U}(\mathsf{goal}_i(k) \wedge \varphi_2))\big)$$

$$\widetilde{\psi}_- \overset{\text{def}}{=} \bigvee_{0 \leqslant k \leqslant \ell_i} \big(\varphi_2 \wedge \mathsf{goal}_i(k) \wedge (\mathsf{run}_i(k) \,\mathrm{S}(\mathsf{init}_i(k) \wedge \varphi_1))\big)$$

Theorem 10 (Soundness) *For each state s in $\mathcal{M}$:*

$$\Pr^{\min}_{\mathcal{M},s}(\varphi) = \Pr^{\min}_{\widetilde{\mathcal{M}},\widetilde{s}}(\widetilde{\varphi}) \ \textit{and} \ \Pr^{\max}_{\mathcal{M},s_{init}}(\varphi) = \Pr^{\max}_{\widetilde{\mathcal{M}},\widetilde{s}}(\widetilde{\varphi})$$

where $\widetilde{s} = \langle s, f_1^s, \ldots, f_m^s \rangle$.

The proof of the soundness of the transformation can be found in the technical report [5]. Thus, we can rely on well-known model-checking techniques for MDPs and LTL. Most prominent is the automata-based approach that transforms the LTL formula into a deterministic ω-automaton $\mathcal{D}$ and then analyzes the end components of the product of the given MDP and $\mathcal{D}$ (see, e.g., [3, 16]). The worst-case time complexity of this approach is dominated by the generation of a deterministic automaton for the LTL formula and runs in time polynomial in the size of the MDP and double exponential in the length of the LTL formula.

In our case, the size of the generated MDP $\widetilde{\mathcal{M}}$ is polynomial in the size of $\mathcal{M}$, but (single) exponential in the length of the longest runs in the automata of subformulas $\ominus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ or $\oplus^{\mathcal{A}}(\varphi_1; \mathsf{constr}; \varphi_2)$ (see Remark 9). Thus, the time complexity of our algorithm is double exponential, too.

4.2 Complexity

We now discuss the complexity of the model-checking problem for $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$ and its sublogic $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Window}]$ over WMDPs, WMCs and WTSs (see Notation 6).

The model-checking problem for standard LTL is known to be 2EXPTIME-complete for MDPs and PSPACE-complete for Markov chains and transition systems [16, 25, 27]. Obviously, the lower bounds carry over to any logic that extends LTL. Nondeterministic polynomially space-bounded algorithms for the model-checking problem for $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$ in WMCs and WTSs arise by adapting the approaches of [25] and [27]. Hence:

Theorem 11 *The model-checking problem for* $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$ *and* $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Window}]$ *is* 2EXPTIME*-complete for WMDPs and* PSPACE*-complete for WMCs and WTSs.*

With the reduction presented in the previous section, the probabilistic analysis has to be carried out with the MDP $\widetilde{\mathcal{M}} = \mathit{Monitor}(\mathcal{M}, \ldots)$ whose size grows exponentially in the sizes of (more precisely, the length of longest runs in) the automata $\mathcal{A} \in \mathsf{AUT}$ that appear as parameters of the modalities $\diamondminus^{\mathcal{A}}$ and $\diamondplus^{\mathcal{A}}$. However, we cannot expect much more efficient algorithms for the $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$-PMC problem since even simple patterns of $\mathrm{PL}[\diamondplus : \mathsf{Window}]$-formulas interpreted over WTSs and WMCs can encode NP-hard problems as shown in the proof of the following theorem.

Theorem 12 (NP/coNP-completeness for WTSs) *For WTSs the problem "does $s_{init} \models \Phi_i$ hold?" is* NP*-complete for formulas of the type Φ_1, Φ_2, Φ_3 and* coNP*-complete for Φ_4, Φ_5 and Φ_6 where:*

$$\Phi_1 = \exists \diamondplus^{=\ell}\, \mathsf{constr} \qquad \Phi_4 = \forall \diamondplus^{=\ell}\, \mathsf{constr}$$
$$\Phi_2 = \exists \Diamond \diamondplus^{=\ell}\, \mathsf{constr} \qquad \Phi_5 = \forall \Diamond \diamondplus^{=\ell}\, \mathsf{constr}$$
$$\Phi_3 = \exists \Box \Diamond \diamondplus^{=\ell}\, \mathsf{constr} \qquad \Phi_6 = \forall \Box \Diamond \diamondplus^{=\ell}\, \mathsf{constr}$$

The same holds when $\diamondplus^{=\ell}$ is replaced with $\diamondplus^{\leqslant \ell}$.

The hardness results in Theorem 12 can be shown using reductions from the subset sum problem and its complement. The coNP-hardness proof for $\diamondplus^{\leqslant \ell}$ uses two positive reward functions. In the remaining cases, the hardness already holds for a WTS with a single positive integer-valued reward function and when constr is a simple basic weight constraint $\mathsf{wgt} = c$ or its negation. Analogous results are obtained for Markov chains, extending the NP-hardness result of [23] for the quantitative PMC decision problem and reward-bounded reachability:

Theorem 13 (NP/coNP-completeness for WMCs) *For WMCs the problems to decide whether*

$$\mathrm{Pr}_{\mathcal{M}, s_{init}}\big(\diamondplus^{=\ell}\, \mathsf{constr}\big) > 0$$
$$\mathrm{Pr}_{\mathcal{M}, s_{init}}\big(\Diamond \diamondplus^{=\ell}\, \mathsf{constr}\big) > 0 \qquad \mathrm{Pr}_{\mathcal{M}, s_{init}}\big(\Diamond \diamondplus^{=\ell}\, \mathsf{constr}\big) = 1$$
$$\mathrm{Pr}_{\mathcal{M}, s_{init}}\big(\Box \Diamond \diamondplus^{=\ell}\, \mathsf{constr}\big) > 0 \qquad \mathrm{Pr}_{\mathcal{M}, s_{init}}\big(\Box \Diamond \diamondplus^{=\ell}\, \mathsf{constr}\big) = 1$$

are NP*-complete.* NP*-hardness even holds with a single positive integer-valued reward function and if constr has the form $\mathsf{wgt} = c$. The problem "does $\mathrm{Pr}_{\mathcal{M}, s_{init}}\big(\diamondplus^{=\ell}\, \mathsf{constr}\big) = 1$ hold?" is* coNP*-complete. The same holds when $\diamondplus^{=\ell}$ is replaced with $\diamondplus^{\leqslant \ell}$.*

So far we presented hardness results for simple patterns of formulas with monitors $\mathcal{A} \in \mathsf{Window}$, partly with simple weight assertions and single positive reward functions. But even for Boolean combinations of simple window weight assertations, the model-checking problem is computationally hard.

Theorem 14 *For* $\mathrm{PL}[\diamondplus : \mathsf{Acyc}]$ *and* $\mathrm{PL}[\diamondplus : \mathsf{Window}]$*, the positive model-checking problem for WMDPs and WMCs and the existential model-checking problem for WTSs are* NP*-complete. Hardness already holds for a single positive integer-valued reward function.*

4.3 Special algorithms for selected formula patterns

As a consequence of Theorem 12, the task to compute $\mathrm{Pr}^{\max}_{\mathcal{M}, s_{init}}(\Diamond \diamondplus^{\mathcal{A}}\, \mathsf{constr})$ for WMDPs with a single reward functions is computationally hard if constr is a conjunctive weight constraint $\mathsf{wgt} = c$ (which is $(\mathsf{wgt} \leqslant c) \wedge (\mathsf{wgt} \geqslant c)$). However, the analogous problem for simple basic weight constraints $\mathsf{wgt} \bowtie c$ can be solved efficiently, even for WMDP with a (possibly negative) weight function.

Proposition 15 *For WMDPs with a single weight function, the problems*

$$\text{"does } \mathrm{Pr}^{\max}_{\mathcal{M}, s_{init}}\big(\Diamond \diamondplus^{\mathcal{A}}(\mathsf{wgt} \bowtie c)\big) > 0 \text{ hold?"}$$
$$\text{"does } \mathrm{Pr}^{\min}_{\mathcal{M}, s_{init}}\big(\Box \boxplus^{\mathcal{A}}(\mathsf{wgt} \bowtie c)\big) = 1 \text{ hold?"}$$

are in P.

Proof. It suffices to consider the problem to decide whether the probability of $\Diamond \diamondplus^{\mathcal{A}}(\mathsf{wgt} \bowtie c)$ is positive as:

$$\mathrm{Pr}^{\min}_{\mathcal{M}, s_{init}}\big(\Box \boxplus^{\mathcal{A}}(\mathsf{wgt} \bowtie c)\big) = 1 - \mathrm{Pr}^{\max}_{\mathcal{M}, s_{init}}\big(\Diamond \diamondplus^{\mathcal{A}}(\mathsf{wgt} \not\bowtie c)\big)$$

We define $\mathit{Graph}[\mathcal{M} \otimes \mathcal{A}]$ as a weighted directed graph with the vertices $\langle s, q \rangle \in S \times Q$ and the following edge relation E:

$$\begin{aligned} & (\langle s, q \rangle, \langle s', q' \rangle) \in E \\ \text{iff} \quad & P(s, \mathit{act}, s') > 0 \text{ for some } \mathit{act} \in \mathit{Act}(s) \\ & \text{and } q' = \delta(q, s') \neq \bot \end{aligned}$$

The weight of the edge from vertex $\langle s, q \rangle$ to vertex $\langle s', q' \rangle$ is

$$\min\big\{\mathit{wgt}(s, \mathit{act}) : P(s, \mathit{act}, s') > 0\big\}.$$

We may apply standard polynomial-time shortest path algorithms (e.g., the algorithms by Bellman-Ford or Floyd) to compute the lengths $\ell_{\min}(s)$ and $\ell_{\max}(s)$ of shortest and longest paths from $\langle s, q_{init} \rangle$ to some state $\langle t, p \rangle$ with $p \in F$ in $\mathit{Graph}[\mathcal{M} \otimes \mathcal{A}]$. Here, the notion of length is to be understood in terms of accumulated weight. (Longest paths are obtained by shortest path algorithms for the weighted graph that results from $\mathit{Graph}[\mathcal{M} \otimes \mathcal{A}]$ by multiplying all weights by -1.) Here, we deal with $\ell_{\min}(s) = +\infty$ if no state in $S \times F$ is reachable from $\langle s, q_{init} \rangle$ and $\ell_{\min}(s) = -\infty$ if some cycle with negative weight is reachable from $\langle s, q_{init} \rangle$. Similarly, we have $\ell_{\max}(s) \in \mathbb{Q} \cup \{-\infty, +\infty\}$ with the same conditions for $-\infty$ and $+\infty$.

Let $\lhd \in \{\leqslant, <\}$ and $\rhd \in \{\geqslant, >\}$. The statement follows from the fact that there is a scheduler $\mathfrak{S}$ for $\mathcal{M}$ where $\mathrm{Pr}^{\mathfrak{S}}_{\mathcal{M}, s_{init}}(\Diamond \diamondplus^{\mathcal{A}}(\mathsf{wgt} \lhd c))$ is positive if and only if there exists a state s in $\mathcal{M}$ that is reachable from s_{init} with $\ell_{\min}(s) \lhd c$. Similarly, there is a scheduler $\mathfrak{S}$ for $\mathcal{M}$ where $\mathrm{Pr}^{\mathfrak{S}}_{\mathcal{M}, s_{init}}(\Diamond \diamondplus^{\mathcal{A}}(\mathsf{wgt} \rhd c))$ is positive if and only if there exists a state s in $\mathcal{M}$ that is reachable from s_{init} with $\ell_{\max}(s) \rhd c$. ∎

According to Theorem 13, the computation of $\mathrm{Pr}_{\mathcal{M}, s_{init}}(\Box \Diamond \diamondplus^{\mathcal{A}}\, \mathsf{constr})$ is hard in WMCs, even in the 1-dimensional case. The problem becomes considerably simpler for basic weight constraints:

Proposition 16 *For WMCs with a single weight function, the probabilities*

$$\mathrm{Pr}_{\mathcal{M}, s_{init}}\big(\Box \Diamond \diamondplus^{\mathcal{A}}(\mathsf{wgt} \bowtie c)\big)$$
$$\mathrm{Pr}_{\mathcal{M}, s_{init}}\big(\Diamond \Box \boxplus^{\mathcal{A}}(\mathsf{wgt} \bowtie c)\big)$$

can be computed in polynomial time.

	General	Non-negative weight functions, simple weight constraints
$\mathrm{PL}[\diamondplus : \mathsf{Acyc}]$ $\mathrm{PL}[\diamondplus : \mathsf{Window}]$	**NP-complete** (Thm. 14)	
$\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$ $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Window}]$	WTS, WMC: **PSPACE-complete** (Thm. 11) WMDP: **2EXPTIME-complete** (Thm. 11)	
$\mathrm{PL}[\diamondplus : \mathsf{Reach}]$ $\mathrm{PL}[\diamondplus : \mathsf{All}]$	**undecidable** (Thm. 17,20)	*decidable*
$\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Reach}]$ $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{All}]$	*undecidable*	**decidable** (Thm. 18)

Table 1. Decidability and complexity of the model-checking problem.

The proof relies on the computation of the bottom strongly connected components that are *good* according to the weight constraint $\mathsf{wgt} \bowtie c$, i.e. those BSCCs T for which there is no finite path π in T with $trace(\pi) \in \mathcal{L}(\mathcal{A})$ and $wgt(\pi \not\bowtie c)$. Checking whether a BSCC is good can be done using shortest path algorithms. Details can be found in the technical report [5].

5. Unbounded weight assertions

So far, we studied the model-checking problem for $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Acyc}]$ where the operators $\diamondminus^{\mathcal{A}}$ and $\diamondplus^{\mathcal{A}}$ are parametrized by acyclic DFA, i.e., their accepted languages are finite. Dropping this assumption leads to undecidability. This is an immediate consequence of the undecidability results by [8] for LTL with prefix-accumulation assertions which can be seen as a sublogic of $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{All}]$; see Section 3.4.

More interesting is the observation that undecidability even holds for the logic $\mathrm{PL}[\diamondplus : \mathsf{Reach}]$, i.e., propositional logic where the atoms are pure weight assertions $\diamondplus^{\mathcal{A}[\ldots\phi]}\,\mathsf{constr}$ where ϕ is an ordinary propositional formula with atoms in AP. Recall that the path conditions specified by the automata $\mathcal{A}[\ldots\phi] \in \mathsf{Reach}$ are reachability constraints $\Diamond\phi$. Given the fact that Boker et al [8] prove decidability for the branching-time logic obtained by adding the CTL-modality $\exists\Diamond$ and prefix-accumulation assertions to propositional logic, this result appears surprising to us. Using a reduction from the Post correspondence problem, we get:

Theorem 17 (Undecidability for $\mathrm{PL}[\diamondplus : \mathsf{Reach}]$) *The model-checking problem for* $\mathrm{PL}[\diamondplus : \mathsf{Reach}]$ *over WTSs and WMCs is undecidable.*

A detailed proof can be found in the technical report [5]. By Theorem 17, there is no chance to design algorithms for the computation of (maximal or minimal) probabilities for the events specified as formulas of $\mathrm{PL}[\diamondplus : \mathsf{Reach}]$ (or more expressive logics such as $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{Reach}]$ and $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{All}]$) in WMCs and WMDPs. We now discuss the case of non-negative structures where all weight functions are non-negative.

Recall that simple weight constraints are Boolean combinations of simple basic weight constraints $\mathsf{wgt}_i \bowtie c$ and that $\mathrm{LTL}_{\mathrm{simple}}[\diamondplus, \diamondminus : \mathsf{All}]$ is the sublogic of $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{All}]$ where all weight constraints are simple.

Theorem 18 *The* $\mathrm{LTL}_{\mathrm{simple}}[\diamondplus, \diamondminus : \mathsf{All}]$*-PMC problem is decidable for non-negative WMDPs.*

The proof can be found in the technical report [5] and relies on an adaption of the algorithm presented in Section 4.1 using a threshold technique that avoids the expansion of pairs $f_i(k) = (q, \overline{w})$ where the values of $\overline{w}$ are larger than the largest constant in the weight constraints of the given formula. To treat 0-weight cycles we use the fact that as soon as $f_i(k) = f_i(k')$ then we can release one argument k or k' and reuse it for fresh runs.
For $d = 1$, all weight constraints are equivalent to simple ones:

Corollary 19 *The* $\mathrm{LTL}[\diamondplus, \diamondminus : \mathsf{All}]$*-PMC problem is decidable for WMDPs with a single non-negative weight function.*

In the multi-dimensional case the requirement that the basic weight expressions are simple cannot be dropped as we have:

Theorem 20 *The model-checking problem for* $\mathrm{PL}[\diamondplus : \mathsf{Reach}]$ *and non-negative WTSs is undecidable. Likewise, the model-checking problem for* $\mathrm{PL}[\diamondplus : \mathsf{Reach}]$ *and non-negative WMCs is undecidable.*

The proof is by a reduction from the model-checking problem for $\mathrm{PL}[\diamondplus : \mathsf{Reach}]$ (see Theorem 17). The idea of the proof is to split each weight function wgt into two non-negative weight functions wgt^+ and wgt^- with $wgt^+ - wgt^- = wgt$, i.e.,

$$\begin{aligned} wgt^+(s, act) &= \max\{wgt(s, act), 0\} \\ wgt^-(s, act) &= -\min\{wgt(s, act), 0\} \end{aligned}$$

for each state s and action act of the WMDP. Then, each appearance of wgt in a weight expression can be replaced by $(\mathsf{wgt}^+ - \mathsf{wgt}^-)$.

6. Conclusions

We established sharp complexity bounds and investigated the border of decidability for the model-checking problem of our new logics. Our main results are depicted in table 4.2, where we distinguish between the general case with arbitrary (rational) weight functions and weight assertions and the case of simple weight assertions with non-negative weight functions. Note that the second column is only applicable if both of these restrictions apply. The given results refer to the positive model-checking problem for WMCs and WMDPs and the existential one for WTSs. The results for the automata class Acyc also hold for the automata class Window. Similarly, the results for Reach also hold for All. The decidability results in the table written in italic are a direct consequence of some result in boldface.

Even though the stated complexity bounds seem to make a practical application unfeasible, there are many techniques to make LTL model checking for MDPs applicable to real-world scenarios. An evaluation of the methods used in some popular model-checking tools can be found, e.g., in [22] for PRISM, in [21] for MRMC and in [6] for ProbDiVinE.

Acknowledgements We are grateful to the operating systems group, especially to Hermann Härtig and Marcus Völp, for providing inspiration for this work through real-world problems. We would also like to thank Marcus Daum, Clemens Dubslaff, Daniel Krähmann, Linda Leuschner, Steffen Märcker and David Müller for fruitful discussions within our group.

References

[1] P. A. Abdulla, R. Mayr, A. Sangnier, and J. Sproston. Solving parity games on integer vectors. In *24th International Conference on Concurrency Theory (CONCUR)*, volume 8052 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2013.

[2] S. Andova, H. Hermanns, and J.-P. Katoen. Discrete-time rewards model-checked. In *First International Workshop on Formal Modeling and Analysis of Timed Systems Workshop (FORMATS)*, volume 2791 of *Lecture Notes in Computer Science*, pages 88–104. Springer, 2003.

[3] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[4] C. Baier, L. Cloth, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Performability assessment by model checking of Markov reward models. *Formal Methods in System Design*, 36(1):1–36, 2010.

[5] C. Baier, J. Klein, S. Klüppelholz, and S. Wunderlich. Weight monitoring with linear temporal logic: Complexity and decidability. Technical report, TU Dresden, 2014. See `http://wwwtcs.inf.tu-dresden.de/ALGI/PUB/CSL-LICS14/`.

[6] J. Barnat, L. Brim, I. Cema, M. Ceska, and J. Tumova. ProbDiVinE: A parallel qualitative ltl model checker. *4th International Conference on Quantitative Evaluation of SysTems (QEST)*, pages 215–216, 2007.

[7] R. Bloem, K. Chatterjee, T. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *21st International Conference on Computer Aided Verification (CAV)*, volume 5643 of *Lecture Notes in Computer Science*, pages 140–156. Springer, 2009. .

[8] U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. In *26th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 43–52. IEEE Computer Society, 2011.

[9] T. Brázdil, V. Brozek, K. Chatterjee, V. Forejt, and A. Kucera. Two views on multiple mean-payoff objectives in Markov decision processes. In *26th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 33–42. IEEE Computer Society, 2011. ISBN 978-0-7695-4412-0.

[10] K. Chatterjee and L. Doyen. Energy and mean-payoff parity Markov decision processes. In *36th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 6907 of *Lecture Notes in Computer Science*, pages 206–218. Springer, 2011.

[11] K. Chatterjee and L. Doyen. Energy parity games. *Theoretical Computer Science*, 458:49–60, 2012.

[12] K. Chatterjee, R. Majumdar, and T. A. Henzinger. Markov decision processes with multiple objectives. In *23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 3884 of *Lecture Notes in Computer Science*, pages 325–336. Springer, 2006. ISBN 978-3-540-32301-3. .

[13] K. Chatterjee, T. Henzinger, B. Jobstmann, and R. Singh. Measuring and synthesizing systems in probabilistic environments. In *22nd International Conference on Computer Aided Verification (CAV)*, volume 6174 of *Lecture Notes in Computer Science*, pages 380–395. Springer, 2010. .

[14] K. Chatterjee, L. Doyen, M. Randour, and J.-F. Raskin. Looking at mean-payoff and total-payoff through windows. In *11th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, pages 118–132, 2013.

[15] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

[16] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

[17] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, Department of Computer Science, 1997.

[18] L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *13th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 454–465. IEEE Computer Society, 1998.

[19] K. Etessami, M. Kwiatkowska, M. Vardi, and M. Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4(4), 2008.

[20] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated verification techniques for probabilistic systems. In *11th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM)*, volume 6659 of *Lecture Notes in Computer Science*, pages 53–113. Springer, 2011.

[21] J.-P. Katoen, I. Zapreev, E. Hahn, H. Hermanns, and D. Jansen. The ins and outs of the probabilistic model checker MRMC. *Performance Evaluation*, 68(2):90–104, 2011.

[22] M. Kwiatkowska, G. Norman, and D. Parker. Advances and challenges of probabilistic model checking. In *48th Annual Allerton Conference on Communication, Control and Computing*, pages 1691–1698. IEEE Press, 2010.

[23] F. Laroussinie and J. Sproston. Model checking durational probabilistic systems. In *8th International Conference on Foundations of Software Science and Computational Structures (FOSSACS)*, volume 3441 of *Lecture Notes in Computer Science*, pages 140–154. Springer, 2005.

[24] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[25] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logic. *Journal of the ACM*, 32(3):733–749, 1985.

[26] T. Tomita, S. Hiura, S. Hagihara, and N. Yonezaki. A temporal logic with mean-payoff constraints. In *14th International Conference on Formal Engineering Methods. Formal Methods and Software Engineering (ICFEM)*, volume 7635 of *Lecture Notes in Computer Science*, pages 249–265. Springer, 2012.

[27] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *1st Symposium on Logic in Computer Science (LICS)*, pages 332–344. IEEE Computer Society Press, 1986.

[28] C. von Essen and B. Jobstmann. Synthesizing systems with optimal average-case behavior for ratio objectives. In *International Workshop on Interactions, Games and Protocols (iWIGP)*, volume 50 of *EPTCS*, pages 17–32, 2011.