Formalizing and Reasoning about Quality [⋆]

Shaull Almagor[1], Udi Boker[2], and Orna Kupferman[1]

[1] The Hebrew University, Jerusalem, Israel.
[2] IST Austria, Klosterneuburg, Austria.

Abstract. Traditional formal methods are based on a Boolean satisfaction notion: a reactive system satisfies, or not, a given specification. We generalize formal methods to also address the *quality* of systems. As an adequate specification formalism we introduce the linear temporal logic $\text{LTL}[\mathcal{F}]$. The satisfaction value of an $\text{LTL}[\mathcal{F}]$ formula is a number between 0 and 1, describing the quality of the satisfaction. The logic generalizes traditional LTL by augmenting it with a (parameterized) set $\mathcal{F}$ of arbitrary functions over the interval $[0, 1]$. For example, $\mathcal{F}$ may contain the maximum or minimum between the satisfaction values of subformulas, their product, and their average.
The classical decision problems in formal methods, such as satisfiability, model checking, and synthesis, are generalized to search and optimization problems in the quantitative setting. For example, model checking asks for the quality in which a specification is satisfied, and synthesis returns a system satisfying the specification with the highest quality. Reasoning about quality gives rise to other natural questions, like the distance between specifications. We formalize these basic questions and study them for $\text{LTL}[\mathcal{F}]$. By extending the automata-theoretic approach for LTL to a setting that takes quality into an account, we are able to solve the above problems and show that reasoning about $\text{LTL}[\mathcal{F}]$ has roughly the same complexity as reasoning about traditional LTL.

1 Introduction

One of the main obstacles to the development of complex computerized systems lies in ensuring their correctness. Efforts in this direction include *temporal-logic model checking* – given a mathematical model of the system and a temporal-logic formula that specifies a desired behavior of the system, decide whether the model satisfies the formula, and *synthesis* – given a temporal-logic formula that specifies a desired behavior, generate a system that satisfies the specification with respect to all environments [6].

Correctness is Boolean: a system can either satisfy its specification or not satisfy it. The richness of today's systems, however, justifies specification formalisms that are *multi-valued*. The multi-valued setting arises directly in systems in which components are multi-valued (c.f., probabilistic and weighted systems) and arises indirectly in applications where multi values are used in order to model missing, hidden, or varying information

[⋆] This work was supported in part by the Austrian Science Fund NFN RiSE (Rigorous Systems Engineering), by the ERC Advanced Grant QUAREM (Quantitative Reactive Modeling), and the ERC Grant QUALITY.
The full version is available at the authors' URLs.

(c.f., abstraction, query checking, and inconsistent viewpoints). As we elaborate below, the multi-valued setting has been an active area of research in recent years. No attempts, however, have been made to augment temporal logics with a quantitative layer that would enable the specification of the relative merits of different aspects of the specification and would enable to formalize the *quality* of a reactive system. Given the growing role that temporal logic plays in planning and robotics, and the criticality of quality in these applications [16], such an augmentation is of great importance also beyond the use of temporal logic in system design and verification.

In this paper we suggest a framework for formalizing and reasoning about quality. Our working assumption is that satisfying a specification is not a yes/no matter. Different ways of satisfying a specification should induce different levels of quality, which should be reflected in the output of the verification procedure. Consider for example the specification $\mathsf{G}(req \rightarrow \mathsf{F}grant)$. There should be a difference between a computation that satisfies it with grants generated soon after requests, one that satisfies it with long waits, one that satisfies it with several grants given to a single request, one that satisfies it vacuously (with no requests), and so on. Moreover, we may want to associate different levels of importance to different components of a specification, to express their mutual influence on the quality, and to formalize the fact that we have different levels of confidence about some of them.

Quality is a rather subjective issue. Technically, we can talk about the quality of satisfaction of specifications since there are different ways to satisfy specifications. We introduce and study the linear temporal logic $\mathrm{LTL}[\mathcal{F}]$, which extends LTL with an arbitrary set $\mathcal{F}$ of functions over $[0,1]$. Using the functions in $\mathcal{F}$, a specifier can formally and easily prioritize the different ways of satisfaction. The logic $\mathrm{LTL}[\mathcal{F}]$ is really a family of logics, each parameterized by a set $\mathcal{F} \subseteq \{f : [0,1]^k \rightarrow [0,1] \mid k \in \mathbb{N}\}$ of functions (of arbitrary arity) over $[0,1]$. For example, $\mathcal{F}$ may contain the $\min\{x,y\}$, $\max\{x,y\}$, and $1-x$ functions, which are the standard quantitative analogues of the $\wedge$, $\vee$, and $\neg$ operators. As we discuss below, such extensions to LTL have already been studied in the context of quantitative verification [15]. The novelty of $\mathrm{LTL}[\mathcal{F}]$, beyond its use in the specification of quality, is the ability to manipulate values by arbitrary functions. For example, $\mathcal{F}$ may contain the quantitative operator $\triangledown_\lambda$, for $\lambda \in [0,1]$, that tunes down the quality of a sub-specification. Formally, the quality of the satisfaction of the specification $\triangledown_\lambda \varphi$ is the multiplication of the quality of the satisfaction of φ by λ. Another useful operator is the weighted-average function $\oplus_\lambda$. There, the quality described by the formula $\varphi \oplus_\lambda \psi$ is the weighted (according to λ) average between the quality of φ and that of ψ. This enables the quality of the system to be an interpolation of different aspects of it. As an example, consider the formula $\mathsf{G}(req \rightarrow (grant \oplus_{\frac{3}{4}} \mathsf{X}grant))$. The formula specifies the fact that we want requests to be granted immediately and the grant to hold for two transactions. When this always holds, the satisfaction value is 1. We are quite okay with grants that are given immediately and last for only one transaction, in which case the satisfaction value is $\frac{3}{4}$, and less content when grants arrive with a delay, in which case the satisfaction value is $\frac{1}{4}$.

An $\mathrm{LTL}[\mathcal{F}]$ formula maps computations to a value in $[0,1]$. We accordingly generalize classical decision problems, such as model checking, satisfiability, synthesis, and equivalence, to their quantitative analogues, which are search or optimization problems.

For example, the equivalence problem between two $\mathrm{LTL}[\mathcal{F}]$ formulas φ_1 and φ_2 seeks the supremum of the difference in the satisfaction values of φ_1 and φ_2 over all computations. Of special interest is the extension of the synthesis problem. In conventional synthesis algorithms we are given a specification to a reactive system, typically by means of an LTL formula, and we transform it into a system that is guaranteed to satisfy the specification with respect to all environments [23]. Little attention has been paid to the quality of the systems that are automatically synthesized[3]. Current efforts to address the quality challenge are based on enriching the game that corresponds to synthesis to a weighted one [2, 5]. Using $\mathrm{LTL}[\mathcal{F}]$, we are able to embody quality within the specification, which is very convenient.

In the Boolean setting, the automata-theoretic approach has proven to be very useful in reasoning about LTL specifications. The approach is based on translating LTL formulas to nondeterministic Büchi automata on infinite words [25]. In the quantitative approach, it seems natural to translate formulas to *weighted automata* [21]. However, these extensively-studied models are complicated and many problems become undecidable for them [1, 17]. We show that we can use the approach taken in [15], bound the number of possible satisfaction values of $\mathrm{LTL}[\mathcal{F}]$ formulas, and use this bound in order to translate $\mathrm{LTL}[\mathcal{F}]$ formulas to Boolean automata. From a technical point of view, the big challenge in our setting is to maintain the simplicity and the complexity of the algorithms for LTL, even though the number of possible values is exponential. We do so by restricting attention to feasible combinations of values assigned to the different subformulas of the specification. Essentially, our translation extends the construction of [25] by associating states of the automaton with functions that map each subformula to a satisfaction value. Using the automata-theoretic approach, we solve the basic problems for $\mathrm{LTL}[\mathcal{F}]$ within the same complexity classes as the corresponding problems in the Boolean setting (as long as the functions in $\mathcal{F}$ are computable within these complexity classes; otherwise, they become the computational bottleneck). Our approach thus enjoys the fact that traditional automata-based algorithms are susceptible to well-known optimizations and symbolic implementations. It can also be easily implemented in existing tools.

Recall that our main contribution is the ability to address the issue of quality within the specification formalism. While we describe it with respect to Boolean systems, we show in Section 5 that our contribution can be generalized to reason about weighted systems, where the values of atomic propositions are taken from $[0, 1]$. We also extend $\mathrm{LTL}[\mathcal{F}]$ to the branching temporal logic $\mathrm{CTL}^{\star}[\mathcal{F}]$, which is the analogous extension of $\mathrm{CTL}^{\star}$, and show that we can still solve decision and search problems. Finally, we define a fragment, $\mathrm{LTL}^{\triangledown}$, of $\mathrm{LTL}[\mathcal{F}]$ for which the number of different satisfaction values is linear in the length of the formula, leading to even simpler algorithms.

Related Work. In recent years, the quantitative setting has been an active area of research, providing many works on quantitative logics and automata [9, 10, 12, 18].

Conceptually, our work aims at formalizing quality, having a different focus from each of the other works. Technically, the main difference between our setting and most

[3] Note that we do not refer here to the challenge of generating optimal (say, in terms of state space) systems, but rather to quality measures that refer to how the specification is satisfied.

of the other approaches is the source of quantitativeness: There, it stems from the nature of the system, whereas in our setting it stems from the richness of the new functional operators. For example, in *multi-valued systems*, the values of atomic propositions are taken from a finite domain [4, 18]. In *fuzzy temporal logic* [22], the atomic propositions take values in $[0,1]$. *Probabilistic temporal logic* is interpreted over Markov decision processes [8, 20], and in the context of *real-valued signals* [11], quantitativeness stems from both time intervals and predicates over the value of atomic propositions.

Closer to our approach is [7], where CTL is augmented with discounting and weighted-average operators. Thus, a formula has a rich satisfaction value, even on Boolean systems. The motivation in [7] is to suggest a logic whose semantics is not too sensitive to small perturbations in the model. Accordingly, formulas are evaluated on weighted-system (as we do in Section 5) or on Markov-chains. We, on the other hand, aim at specifying quality of on-going behaviors. Hence, we work with the much stronger LTL and CTL* logics, and we augment them by arbitrary functions over $[0,1]$.

A different approach, orthogonal to ours, is to stay with Boolean satisfaction values, while handling quantitative properties of the system, in particular ones that are based on unbounded accumulation [3]. The main challenge in these works is the border of decidability, whereas our technical challenge is to keep the simplicity of the algorithms known for LTL in spite of the exponential number of satisfaction values. Nonetheless, an interesting future research direction is to combine the two approaches.

2 Formalizing Quality

2.1 The Temporal Logic LTL$[\mathcal{F}]$

The linear temporal logic LTL$[\mathcal{F}]$ generalizes LTL by replacing the Boolean operators of LTL with arbitrary functions over $[0,1]$. The logic is actually a family of logics, each parameterized by a set $\mathcal{F}$ of functions.

Syntax. Let AP be a set of Boolean atomic propositions, and let $\mathcal{F} \subseteq \{f : [0,1]^k \to [0,1] \mid k \in \mathbb{N}\}$ be a set of functions over $[0,1]$. Note that the functions in $\mathcal{F}$ may have different arities. An LTL$[\mathcal{F}]$ formula is one of the following:

- True, False, or p, for $p \in AP$.
- $f(\varphi_1, ..., \varphi_k)$, $\mathsf{X}\varphi_1$, or $\varphi_1 \mathsf{U} \varphi_2$, for LTL$[\mathcal{F}]$ formulas $\varphi_1, \ldots, \varphi_k$ and a function $f \in \mathcal{F}$.

Semantics. The semantics of LTL$[\mathcal{F}]$ formulas is defined with respect to (finite or infinite) computations over AP. We use $(2^{AP})^\infty$ to denote $(2^{AP})^* \cup (2^{AP})^\omega$. A *computation* is a word $\pi = \pi_0, \pi_1, \ldots \in (2^{AP})^\infty$. We use π^i to denote the suffix $\pi_i, \pi_{i+1}, \ldots$. The semantics maps a computation π and an LTL$[\mathcal{F}]$ formula φ to the *satisfaction value* of φ in π, denoted $[\![\pi, \varphi]\!]$. The satisfaction value is defined inductively as described in Table 1 below.[4]

[4] The observant reader may be concerned by our use of $\max$ and $\min$ where $\sup$ and $\inf$ are in order. In Lemma 1 we prove that there are only finitely many satisfaction values for a formula φ, thus the semantics is well defined.

Formula	Satisfaction value	Formula	Satisfaction value
$[\![\pi, \texttt{True}]\!]$	1	$[\![\pi, f(\varphi_1, ..., \varphi_k)]\!]$	$f([\![\pi, \varphi_1]\!], ..., [\![\pi, \varphi_k]\!])$
$[\![\pi, \texttt{False}]\!]$	0	$[\![\pi, \mathsf{X}\varphi_1]\!]$	$[\![\pi^1, \varphi_1]\!]$
$[\![\pi, p]\!]$	1 if $p \in \pi_0$ 0 if $p \notin \pi_0$	$[\![\pi, \varphi_1 \mathsf{U} \varphi_2]\!]$	$\max\limits_{0 \leq i < \lvert\pi\rvert} \{\min\{[\![\pi^i, \varphi_2]\!], \min\limits_{0 \leq j < i} [\![\pi^j, \varphi_1]\!]\}\}$

Table 1. The semantics of LTL$[\mathcal{F}]$.

It is not hard to prove, by induction on the structure of the formula, that for every computation π and formula φ, it holds that $[\![\pi, \varphi]\!] \in [0, 1]$. We use the usual $\mathsf{F}\varphi_1 = \texttt{True}\mathsf{U}\varphi_1$ and $\mathsf{G}\varphi_1 = \neg(\texttt{True}\mathsf{U}(\neg\varphi_1))$ abbreviations.

The logic LTL coincides with the logic LTL$[\mathcal{F}]$ for $\mathcal{F}$ that corresponds to the usual Boolean operators. For simplicity, we use the common such functions as abbreviation, as described below. In addition, we introduce notations for some useful functions. Let $x, y \in [0, 1]$. Then,

- $\neg x = 1 - x$
- $x \vee y = \max\{x, y\}$
- $x \wedge y = \min\{x, y\}$
- $\nabla_\lambda x = \lambda \cdot x$
- $x \oplus_\lambda y = \lambda \cdot x + (1 - \lambda) \cdot y$

To see that LTL indeed coincides with LTL$[\mathcal{F}]$ for $\mathcal{F} = \{\neg, \vee, \wedge\}$, note that for this $\mathcal{F}$, all formulas are mapped to $\{0, 1\}$ in a way that agrees with the semantics of LTL.

Kripke structures and transducers. For a Kripke structure $\mathcal{K}$ and an LTL$[\mathcal{F}]$ formula φ, we have that $[\![\mathcal{K}, \varphi]\!] = \min\{[\![\pi, \varphi]\!] : \pi \text{ is a computation of } \mathcal{K}\}$. That is, the value is induced by the path that admits the lowest satisfaction value. [5]

In the setting of open systems, the set of atomic propositions is partitioned into sets I and O of input and output signals. An (I, O)-transducer then models the computations generated (deterministically) by the system when it interacts with an environment that generates finite or infinite sequences of input signals.

Example 1. Consider a scheduler that receives requests and generates grants. Consider the LTL$[\mathcal{F}]$ formula $\mathsf{G}(\mathit{req} \to \mathsf{F}(\mathit{grant} \oplus_{\frac{1}{2}} \mathsf{X}\mathit{grant})) \wedge \neg(\nabla_{\frac{3}{4}} \mathsf{G}\neg\mathit{req})$. The satisfaction value of the formula is 1 if every request is eventually granted, and the grant lasts for two consecutive steps. If a grant holds only for a single step, then the satisfaction value is reduced to $\frac{1}{2}$. In addition, if there are no requests, then the satisfaction value is at most $\frac{1}{4}$. This shows how we can embed vacuity tests in the formula.

2.2 The Basic Questions

In the Boolean setting, an LTL formula maps computations to $\{\texttt{True}, \texttt{False}\}$. In the quantitative setting, an LTL$[\mathcal{F}]$ formula maps computations to $[0, 1]$. Classical decision problems, such as model checking, satisfiability, synthesis, and equivalence, are accordingly generalized to their quantitative analogues, which are search or optimization problems. Below we specify the basic questions with respect to LTL$[\mathcal{F}]$. While the

[5] Since a Kripke structure may have infinitely many computations, here too we should have a-priori used $\inf$, and the use of $\min$ is justified by Lemma 1.

definition here focuses on $\mathrm{LTL}[\mathcal{F}]$, the questions can be asked with respect to arbitrary quantitative specification formalism, with the expected adjustments.

- The *satisfiability* problem gets as input an $\mathrm{LTL}[\mathcal{F}]$ formula φ and returns $\max\{[\![\pi, \varphi]\!] : \pi \text{ is a computation}\}$. Dually, the *validity* problem returns, given an $\mathrm{LTL}[\mathcal{F}]$ formula φ, the value $\min\{[\![\pi, \varphi]\!] : \pi \text{ is a computation}\}$. [6]
- The *implication* problem gets as input two $\mathrm{LTL}[\mathcal{F}]$ formulas φ_1 and φ_2 and returns $\max\{[\![\pi, \varphi_1]\!] - [\![\pi, \varphi_2]\!] : \pi \text{ is a computation}\}$. The symmetric version of implication, namely the *equivalence* problem, gets as input two $\mathrm{LTL}[\mathcal{F}]$ formulas φ_1 and φ_2 and returns $\max\{|[\![\pi, \varphi_1]\!] - [\![\pi, \varphi_2]\!]| : \pi \text{ is a computation}\}$.
- The *model-checking* problem is extended from the Boolean setting to find, given a system $\mathcal{K}$ and an $\mathrm{LTL}[\mathcal{F}]$ formula φ, the satisfaction value $[\![\mathcal{K}, \varphi]\!]$.
- The *realizability* problem gets as input an LTL formula over $I \cup O$, for sets I and O of input and output signals, and returns $\max\{[\![\mathcal{T}, \varphi]\!] : \mathcal{T} \text{ is an } (I, O)\text{-transducer}\}$. The *synthesis* problem is then to find a transducer that attains this value.

Decision problems. The above questions are search and optimization problems. It is sometimes interesting to consider the decision problems they induce, when referring to a threshold. For example, the model-checking decision-problem is to decide, given a system $\mathcal{K}$, a formula φ, and a threshold t, whether $[\![\mathcal{K}, \varphi]\!] \geq t$. For some problems, there are natural thresholds to consider. For example, in the implication problem, asking whether $\max\{[\![\pi, \varphi_1]\!] - [\![\pi, \varphi_2]\!] : \pi \text{ is a computation}\} \geq 0$ amounts to asking whether for all computations π, we have that $[\![\pi, \varphi_1]\!] \geq [\![\pi, \varphi_2]\!]$, which indeed captures implication.

2.3 Properties of $\mathrm{LTL}[\mathcal{F}]$

Bounding the number of satisfaction values. For an $\mathrm{LTL}[\mathcal{F}]$ formula φ, let $V(\varphi) = \{[\![\pi, \varphi]\!] : \pi \in (2^{AP})^\infty\}$. That is, $V(\varphi)$ is the set of possible satisfaction values of φ in arbitrary computations. We first show that this set is finite for all $\mathrm{LTL}[\mathcal{F}]$ formulas.

Lemma 1. *For every* $\mathrm{LTL}[\mathcal{F}]$ *formula* φ*, we have that* $|V(\varphi)| \leq 2^{|\varphi|}$.

The good news that follows from Lemma 1 is that every $\mathrm{LTL}[\mathcal{F}]$ formula has only finitely many possible satisfaction values. This enabled us to replace the $\sup$ and $\inf$ operators in the semantics by $\max$ and $\min$. It also implies that we can point to witnesses that exhibit the satisfaction values. However, Lemma 1 only gives an exponential bound to the number of satisfaction values. We now show that this exponential bound is tight.

Example 2. Consider the logic $\mathrm{LTL}[\{\oplus\}]$, augmenting LTL with the average function, where for every $x, y \in [0,1]$ we have that $x \oplus y = \frac{1}{2}x + \frac{1}{2}y$. Let $n \in \mathbb{N}$ and consider the formula $\varphi_n = p_1 \oplus (p_2 \oplus (p_3 \oplus (p_4 \oplus ...p_n))...)$. The length of φ_n is in $O(n)$ and the nesting depth of $\oplus$ operators in it is n. For every computation π it holds that

$$[\![\pi, \varphi_n]\!] = \frac{1}{2}[\![\pi_0, p_1]\!] + \frac{1}{4}[\![\pi_0, p_2]\!] + ... + \frac{1}{2^{n-1}}[\![\pi_0, p_{n-1}]\!] + \frac{1}{2^{n-1}}[\![\pi_0, p_n]\!].$$

[6] Lemma 1 guarantees that $\max$ and $\min$ (rather than $\sup$ and $\inf$) are defined.

Hence, every assignment $\pi_0 \subseteq \{p_1, ..., p_{n-1}\}$ to the first position in π induces a different satisfaction value for $[\![\pi, \varphi_n]\!]$, implying that there are 2^{n-1} different satisfaction values for φ_n.

A Boolean look at LTL[$\mathcal{F}$]. LTL[$\mathcal{F}$] provides means to generalize LTL to a quantitative setting. Yet, one may consider a Boolean logic defined by LTL[$\mathcal{F}$] formulas and predicates. For example, having formulas of the form $\varphi_1 \geq \varphi_2$ or $\varphi_1 \geq v$, for LTL[$\mathcal{F}$] formulas φ_1 and φ_2, and a value $v \in [0, 1]$. It is then natural to compare the expressiveness and succinctness of such a logic with respect to LTL.

One may observe that the role the functions in $\mathcal{F}$ play in LTL[$\mathcal{F}$] is propositional, in the sense that the functions do not introduce new temporal operators. We formalize this intuition in the full version, showing that for every LTL[$\mathcal{F}$] formula φ and predicate $P \subseteq [0, 1]$, there exists an LTL formula $Bool(\varphi, P)$ equivalent to the assertion $\varphi \in P$. Formally, we have the following.

Theorem 1. *For every* LTL[$\mathcal{F}$] *formula φ and predicate $P \subseteq [0, 1]$, there exists an* LTL *formula $Bool(\varphi, P)$, of length at most exponential in φ, such that for every computation π, it holds that $[\![\pi, \varphi]\!] \in P$ iff $\pi \models Bool(\varphi, P)$.*

The translation described in the proof of Theorem 1 may involve an exponential blow-up. We indeed conjecture that this blowup is unavoidable, implying that LTL[$\mathcal{F}$], when used as a Boolean formalism, is exponentially more succinct than LTL. Since very little is known about lower bounds for propositional formulas, we leave it as a conjecture. We demonstrate the succinctness with the following example.

Example 3. For $k \geq 1$, let $\oplus_{\frac{1}{k}}$ be the k-ary average operator. Consider the logic LTL[$\{\oplus_{\frac{1}{k}}\}$], for an even integer k, and consider the formula $\varphi_k = \oplus_{\frac{1}{k}}(p_1, \ldots, p_k)$, for the atomic propositions $p_1, \ldots, p_k$.

For every computation π, it holds that $[\![\pi, \varphi_k]\!] = \frac{|\{i\colon p_i \in \pi_0\}|}{k}$. Hence, $[\![\pi, \varphi_k]\!] = \frac{1}{2}$ iff exactly half of the atomic propositions $p_1, \ldots, p_k$ hold in π_0. We conjecture that the LTL formula $Bool(\varphi_k, \frac{1}{2})$ must be exponential in k. Intuitively, the formula has to refer to every subset of size $\frac{k}{2}$. A naive implementation of this involves $\binom{k}{k/2}$ clauses, which is exponential in k. The question whether this can be done with a formula that is polynomial in k is a long-standing open problem.

3 Translating LTL[$\mathcal{F}$] to Automata

The *automata-theoretic* approach uses the theory of automata as a unifying paradigm for system specification, verification, and synthesis [24, 26]. In this section we describe an automata-theoretic framework for reasoning about LTL[$\mathcal{F}$] specifications. In order to explain our framework, let us recall first the translation of LTL formulas to nondeterministic generalized Büchi automata (NGBW), as introduced in [25]. We start with the definition of NGBWs. An NGBW is $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, where Σ is the input alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, and $\alpha \subseteq 2^Q$ is a set of sets of accepting states. The number of sets in α is the *index* of $\mathcal{A}$. A run $r = r_0, r_1, \cdots$ of $\mathcal{A}$ on a word $w = w_1 \cdot w_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states such that $r_0 \in Q_0$, and for every $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, w_{i+1})$. We denote by $\inf(r)$ the set of states that r visits infinitely often, that is $\inf(r) = \{q : r_i = q \text{ for infinitely many } i \in \mathbb{N}\}$. The run r is *accepting* if it visits all the sets in α infinitely often. Formally, for every set $F \in \alpha$ we have that $\inf(r) \cap F \neq \emptyset$. An automaton accepts a word if it has an accepting run on it. The language of an automaton $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts.

In the Vardi-Wolper translation of LTL formulas to NGBWs [25], each state of the automaton is associated with a set of formulas, and the NGBW accepts a computation from a state q iff the computation satisfies exactly all the formulas associated with q. The state space of the NGBW contains only states associated with maximal and consistent sets of formulas, the transitions are defined so that requirements imposed by temporal formulas are satisfied, and the acceptance condition is used in order to guarantee that requirements that involve the satisfaction of eventualities are not delayed forever.

In our construction here, each state of the NGBW assigns a satisfaction value to every subformula. Consistency then assures that the satisfaction values agree with the functions in $\mathcal{F}$. Similar adjustments are made to the transitions and the acceptance condition. The construction translates an $\mathrm{LTL}[\mathcal{F}]$ formula φ to an NGBW, while setting its initial states according to a required predicate $P \subseteq [0,1]$. We then have that for every computation $\pi \in (2^{AP})^\omega$, the resulting NGBW accepts π iff $[\![\pi, \varphi]\!] \in P$.

We note that a similar approach is taken in [15], where LTL formulas are interpreted over quantitative systems. The important difference is that the values in our construction arise from the formula and the functions it involves, whereas in [15] they are induced by the values of the atomic propositions.

Theorem 2. *Let φ be an $\mathrm{LTL}[\mathcal{F}]$ formula and $P \subseteq [0,1]$ be a predicate. There exists an NGBW $\mathcal{A}_{\varphi,P}$ such that for every computation $\pi \in (2^{AP})^\omega$, it holds that $[\![\pi, \varphi]\!] \in P$ iff $\mathcal{A}_{\varphi,P}$ accepts π. Furthermore, $\mathcal{A}_{\varphi,P}$ has at most $2^{(|\varphi|^2)}$ states and index at most $|\varphi|$.*

Proof. We define $\mathcal{A}_{\varphi,P} = \langle 2^{AP}, Q, \delta, Q_0, \alpha \rangle$ as follows. Let $cl(\varphi)$ be the set of φ's subformulas. Let C_φ be the collection of functions $g : cl(\varphi) \to [0,1]$ such that for all $\psi \in cl(\varphi)$, we have that $g(\psi) \in V(\psi)$. For a function $g \in C_\varphi$, we say that g is *consistent* if for every $\psi \in cl(\varphi)$, the following holds.

- If $\psi = \mathtt{True}$, then $g(\psi) = 1$, and if $\psi = \mathtt{False}$ then $g(\psi) = 0$.
- If $\psi = p \in AP$, then $g(\psi) \in \{0,1\}$.
- If $\psi = f(\psi_1, \ldots, \psi_k)$, then $g(\psi) = f(g(\psi_1), \ldots, g(\psi_k))$.

The state space Q of $\mathcal{A}_{\varphi,P}$ is the set of all consistent functions in C_φ. Then, $Q_0 = \{g \in Q : g(\varphi) \in P\}$ contains all states in which the value assigned to φ is in P.

We now define the transition function δ. For functions g, g' and a letter $\sigma \in \Sigma$, we have that $g' \in \delta(g, \sigma)$ iff the following hold.

- $\sigma = \{p \in AP : g(p) = 1\}$.
- For all $\mathsf{X}\psi_1 \in cl(\varphi)$ we have $g(\mathsf{X}\psi_1) = g'(\psi_1)$.
- For all $\psi_1 \mathsf{U} \psi_2 \in cl(\varphi)$ we have $g(\psi_1 \mathsf{U} \psi_2) = \max\{g(\psi_2), \min\{g(\psi_1), g'(\psi_1 \mathsf{U} \psi_2)\}\}$.

Finally, every formula $\psi_1 \mathsf{U} \psi_2$ contributes to α the set $F_{\psi_1 \mathsf{U} \psi_2} = \{g : g(\psi_2) = g(\psi_1 \mathsf{U} \psi_2)\}$.

Remark 1. The construction described in the proof of Theorem 2 is such that selecting the set of initial states allows us to specify any (propositional) condition regarding the sub-formulas of φ. A simple extension of this idea allows us to consider a set of formulas $\{\varphi_1, ..., \varphi_m\} = \Phi$ and a predicate $P \subseteq [0,1]^m$, and to construct an NGBW that accepts a computation π iff $\langle [\![\pi, \varphi_1]\!], ..., [\![\pi, \varphi_n]\!] \rangle \in P$. Indeed, the state space of the product consists of functions that map all the formulas in Φ to their satisfaction values, and we only have to choose as the initial states these functions g for which $\langle g(\varphi_1), ..., g(\varphi_n) \rangle \in P$. As we shall see in Section 4, this allows us to use the automata-theoretic approach also in order to examine relations between the satisfaction values of different formulas.

4 Solving the Basic Questions for $\mathrm{LTL}[\mathcal{F}]$

In this section we solve the basic questions defined in Section 2.2. We show that they all can be solved for $\mathrm{LTL}[\mathcal{F}]$ with roughly the same complexity as for LTL. When we analyze complexity, we assume that the functions in $\mathcal{F}$ can be computed in a complexity that is subsumed by the complexity of the problem for LTL (PSPACE, except for 2EXPTIME for realizability), which is very reasonable. Otherwise, computing the functions becomes the computational bottleneck. A related technical observation is that, assuming the functions in $\mathcal{F}$ can be calculated in PSPACE, we can also enumerate in PSPACE the set $V(\varphi)$ of the possible satisfaction values of an $\mathrm{LTL}[\mathcal{F}]$ formula φ.

The questions in the quantitative setting are basically search problems, asking for the best or worst value. Since every $\mathrm{LTL}[\mathcal{F}]$ formula may only have exponentially many satisfaction values, one can reduce a search problem to a set of decision problems with respect to specific thresholds, remaining in PSPACE. Combining this with the construction of NGBWs described in Theorem 2 is the key to our algorithms.

We can now describe the algorithms in detail.

Satisfiability and validity. We start with satisfiability and solve the decision version of the problem: given φ and a threshold v, decide whether there exists a computation π such that $[\![\pi, \varphi]\!] \geq v$. The latter can be solved by checking the nonemptiness of the NGBW $\mathcal{A}_{\varphi,P}$ with $P = [v, 1]$. Since the NGBW can be constructed on-the-fly, this can be done in PSPACE in the size of $|\varphi|$. The search version can be solved in PSPACE by iterating over the set of relevant thresholds.

We proceed to validity. It is not hard to see that for all φ and v, we have that $\forall \pi, [\![\pi, \varphi]\!] \geq v$ iff $\neg(\exists \pi,\ [\![\pi, \varphi]\!] < v)$. The latter can be solved by checking, in PSPACE, the nonemptiness of the NGBW $\mathcal{A}_{\varphi,P}$ with $P = [0, v)$. Since PSPACE is closed under complementation, we are done. In both cases, the nonemptiness algorithm can return the witness to the nonemptiness.

Implication and equivalence. In the Boolean setting, implication can be reduced to validity, which is in turn reduced to satisfiability. Doing the same here is more sophisticated, but possible: we add to $\mathcal{F}$ the average and negation operators. It is not hard to verify that for every computation π, it holds that $[\![\pi, \varphi_1 \oplus_{\frac{1}{2}} \neg\varphi_2]\!] = \frac{1}{2}([\![\pi, \varphi_1]\!] - [\![\pi, \varphi_2]\!]) + \frac{1}{2}$. In particular, $\max\{[\![\pi, \varphi_1]\!] - [\![\pi, \varphi_2]\!] :\ \pi$ is a computation$\} = 2 \cdot \max\{[\![\pi, \varphi_1 \oplus_{\frac{1}{2}} \neg\varphi_2]\!] :\ \pi$ is a computation$\} - 1$. Thus, the problem reduces to the satisfiability of $\varphi_1 \oplus_{\frac{1}{2}} \neg\varphi_2$, which is solvable in PSPACE. Note that, alternatively, one can proceed as suggested in Remark 1 and reason about the composition of the NGBWs for φ_1 and φ_2. The solution to the equivalence problem is similar, by checking both directions of the implication.

Model checking. The complement of the problem, namely whether there exists a computation π of $\mathcal{K}$ such that $[\![\pi, \varphi]\!] < v$, can be solved by taking the product of the NGBW $\mathcal{A}_{\varphi,(0,v]}$ from Theorem 2 with the system $\mathcal{K}$ and checking for emptiness on-the-fly. As in the Boolean case, this can be done in PSPACE. Moreover, in case the product is not empty, the algorithm returns a witness: a computation of $\mathcal{K}$ that satisfies φ with a low quality. We note that in the case of a single computation, motivated by multi-valued monitoring [11], one can label the computation in a bottom-up manner, as in CTL model checking, and the problem can be solved in polynomial time.

Realizability and synthesis. Several algorithms are suggested in the literature for solving the LTL realizability problem [23]. Since they are all based on a translation of specifications to automata, we can adopt them. Here we describe an adoption of the Safraless algorithm of [19] and its extension to NGBWs. Given φ and v, the algorithm starts by constructing the NGBW $\mathcal{A}_{\varphi,[0,v)}$ and dualizing it to a universal generalized co-Büchi automaton (UGCW) $\tilde{\mathcal{A}}_{\varphi,[0,v)}$. Since dualization amounts to complementation, $\tilde{\mathcal{A}}_{\varphi,[0,v)}$ accepts exactly all computations π with $[\![\pi, \varphi]\!] \geq v$. Being universal, we can expand $\tilde{\mathcal{A}}_{\varphi,[0,v)}$ to a universal tree automaton $\mathcal{U}$ that accepts a tree with directions in 2^I and labels in 2^O if all its branches, which correspond to input sequences, are labeled by output sequences such that the composition of the input and output sequences is a computation accepted by $\tilde{\mathcal{A}}_{\varphi,[0,v)}$. Realizability then amounts to checking the nonemptiness of $\mathcal{U}$ and synthesis to finding a witness to its nonemptiness. Since φ only has an exponential number of satisfaction values, we can solve the realizability and synthesis search problems by repeating this procedure for all relevant values. Since the size of $\mathcal{A}_{\varphi,[0,v)}$ is single-exponential in $|\varphi|$, the complexity is the same as in the Boolean case, namely 2EXPTIME-complete.

5 Beyond LTL$[\mathcal{F}]$

The logic LTL$[\mathcal{F}]$ that we introduce and study here is a first step in our effort to introduce reasoning about quality to formal methods. Future work includes stronger formalisms and algorithms. We distinguish between extensions that stay in the area of LTL$[\mathcal{F}]$ and ones that jump to the (possibly undecidable) world of infinitely many satisfaction values. In the latter, we include efforts to extend LTL$[\mathcal{F}]$ by temporal operators in which the future is discounted, and efforts to combine LTL$[\mathcal{F}]$ with other qualitative aspects of systems [3]. In this section we describe two extensions of the first class: an extension of LTL$[\mathcal{F}]$ to weighted systems and to a branching-time temporal logic. We also describe a computationally simple fragment of LTL$[\mathcal{F}]$.

Weighted systems. A *weighted Kripke structure* is a tuple $\mathcal{K} = \langle AP, S, I, \rho, L \rangle$, where $AP, S, I,$ and ρ are as in Boolean Kripke structures, and $L : S \to [0, 1]^{AP}$ maps each state to a weighted assignment to the atomic propositions. Thus, the value $L(s)(p)$ of an atomic proposition $p \in AP$ in a state $s \in S$ is a value in $[0, 1]$. The semantics of

$\mathrm{LTL}[\mathcal{F}]$ with respect to a weighted computation coincides with the one for non-weighted systems, except that for an atomic proposition p, we have that $[\![\pi, p]\!] = L(\pi_0)(p)$.

It is not hard to extend the construction of $\mathcal{A}_{\varphi,P}$, as described in the proof of Theorem 2, to an alphabet W^{AP}, where W is a set of possible values for the atomic propositions. Indeed, we only have to adjust the transitions so that there is a transition from state g with letter $\sigma \in W^{AP}$ only if g agrees with σ on the values of the atomic propositions. Hence, in settings where the values for the atomic propositions are known, and in particular model checking, the solutions to the basic questions is similar to the ones described for $\mathrm{LTL}[\mathcal{F}]$ with Boolean atomic propositions.

Formalizing quality with branching temporal logics. Formulas of $\mathrm{LTL}[\mathcal{F}]$ specify on-going behaviors of linear computations. A Kripke structure is not linear, and the way we interpret $\mathrm{LTL}[\mathcal{F}]$ formulas with respect to it is universal. In *branching temporal logic* one can add universal and existential quantifiers to the syntax of the logic, and specifications can refer to the branching nature of the system [13].

The branching temporal logic $\mathrm{CTL}^\star[\mathcal{F}]$ extends $\mathrm{LTL}[\mathcal{F}]$ by the path quantifiers E and A. Formulas of the form $\mathsf{E}\varphi$ and $\mathsf{A}\varphi$ are referred to as *state formulas* and they are interpreted over states s in the structure with the semantics $[\![s, \mathsf{E}\varphi]\!] = \max\{[\![\pi, \varphi]\!] : \pi \text{ starts in } s\}$ and $[\![s, \mathsf{A}\varphi]\!] = \min\{[\![\pi, \varphi]\!] : \pi \text{ starts in } s\}$.

In [14], the authors describe a general technique for extending the scope of LTL model-checking algorithms to CTL*. The idea is to repeatedly consider an innermost state subformula, view it as an (existentially or universally quantified) LTL formula, apply LTL model checking in order to evaluate it in all states, and add a fresh atomic proposition that replaces this subformula and holds in exactly these states that satisfy it. This idea, together with our ability to model check systems with weighted atomic propositions, can be used also for model checking $\mathrm{CTL}^\star[\mathcal{F}]$.

More challenging is the handling of the other basic problems. There, the solution involves a translation of $\mathrm{CTL}^\star[\mathcal{F}]$ formulas to tree automata. Since the automata-theoretic approach for CTL* has the Vardi-Wolper construction at its heart, this is possible.

The fragment $\mathrm{LTL}^\triangledown$ *of* $\mathrm{LTL}[\mathcal{F}]$. In the proof of Lemma 1, we have seen that a formula may take exponentially many satisfaction values. The proof crucially relies on the fact that the value of a function is a function of all its inputs. However, in the case of unary functions, or indeed functions that do not take many possible values, this bound can be lowered. Such an interesting fragment is the logic $\mathrm{LTL}^\triangledown = \mathrm{LTL}[\{\triangledown_\lambda, \blacktriangledown_\lambda\}_{\lambda\in[0,1]} \cup \{\vee, \neg\}]$, with the functions $\triangledown_\lambda(x) = \lambda \cdot x$ and $\blacktriangledown_\lambda(x) = \lambda \cdot x + (1-\lambda)/2$.

This fragment is interesting in two aspects. First, computationally, an $\mathrm{LTL}^\triangledown$ formula has only polynomially many satisfaction values. Moreover, for a predicate of the form $P = [v, 1]$ (resp. $P = (v, 1]$), the LTL formula $\mathit{Bool}(\varphi, P)$ can be shown to be of linear length in $|\varphi|$. This implies that solving threshold-problems for $\mathrm{LTL}^\triangledown$ formulas can be done with tools that work with LTL with no additional complexity. Second, philosophically, an interesting question that arises when formalizing quality regards how the *lack* of quality in a component should be viewed. With quality between 0 and 1, we have that 1 stands for "good", 0 for "bad", and $\frac{1}{2}$ for "not good and not bad". While the $\triangledown_\lambda$ operator enables us to reduce the quality towards "badness", the $\blacktriangledown_\lambda$ operator enables us to do so towards "ambivalence".

References

1. S. Almagor, U. Boker, and O. Kupferman. What's decidable about weighted automata? In *8th ATVA, LNCS 6996*, pages 482–491, 2011.
2. R. Bloem, K. Chatterjee, T.A. Henzinger, and B. Jobstmann. Better Quality in Synthesis through Quantitative Objectives. In *Proc. 21st CAV*, LNCS 5643, pages 140–156, 2009.
3. U. Boker, K. Chatterjee, T.A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. In *26th LICS*, pages 43–52, 2011.
4. G. Bruns and P. Godefroid. Model checking with multi-valued logics. In *Proc. 31st ICALP*, LNCS 3142, pages 281–293, 2004.
5. P. Cerný, K. Chatterjee, T.A. Henzinger, A. Radhakrishna, and R. Singh. Quantitative Synthesis for Concurrent Programs. In *Proc. 23rd CAV*, pages 243–259, 2011.
6. E. Clarke, T.A. Henzinger, and H. Veith. *Handbook of Model Checking*. Elsvier, 2011. Forthcoming.
7. L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga, Model checking discounted temporal properties, *TCS*, 345(1):139–170, 2005.
8. J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled markov processes. *TCS*, 318(3):323–354, 2004.
9. M. Droste, W. Kuich, and G. Rahonis. Multi-valued MSO logics over words and trees. *Fundamenta Informaticae*, 84(3-4):305–327, 2008.
10. M. Droste and G. Rahonis. Weighted automata and weighted logics with discounting. *TCS*, 410(37):3481–3494, 2009.
11. A. Donze, O. Maler, E. Bartocci, D. Nickovic, R. Grosu, and S. Smolka. On Temporal Logic and Signal Processing. In *Proc. 10th ATVA*, 2012.
12. M. Droste, K. Werner, and V. Heiko. *Handbook of Weighted Automata*. Springer, 2009.
13. E.A. Emerson and J.Y. Halpern. Sometimes and not never revisited: On branching versus linear time. *Journal of the ACM*, 33(1):151–178, 1986.
14. E.A. Emerson and C.L. Lei. Modalities for model checking: Branching time logic strikes back. In *Proc. 12th POPL*, pages 84–96, 1985.
15. M. Faella, A. Legay, and M. Stoelinga. Model Checking Quantitative Linear Time Logic. *TCS*, 220(3):61–77, 2008.
16. H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Trans. on Robotics* 25(6): 1370–1381, 2009.
17. D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *International Journal of Algebra and Computation*, 4(3): 405–425, 1994.
18. O. Kupferman and Y. Lustig. Lattice automata. In *Proc. 8th VMCAI*, LNCS 4349, pages 199 – 213, 2007.
19. O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th FOCS*, pages 531–540, 2005.
20. M.Z. Kwiatkowska. Quantitative verification: models techniques and tools. *FSE*, pages 449–458, 2007.
21. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
22. S. Moon, K. H. Lee, and D. Lee. Fuzzy branching temporal logic. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 34(2):1045–1055, 2004.
23. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th POPL*, pages 179–190, 1989.
24. W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 133–191, 1990.
25. M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st LICS*, pages 332–344, 1986.
26. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *I&C*, 115(1):1–37, 1994.