Model checking Quantitative Linear Time Logic

Marco Faella
Università di Napoli “Federico II”, Italy

Axel Legay
University of Liège, Belgium

Mariëlle Stoelinga
University of Twente, The Netherlands

Abstract
This paper considers QLTL, a quantitative analagon of LTL and presents algorithms for model checking QLTL over quantitative versions of Kripke structures and Markov chains.

Keywords: Linear temporal logic, Quantitative verification, Automata.

1 Introduction

Quantitative properties, such as real-time and resource consumption, are essential in embedded system design. Hence, a wide variety of verification frameworks have been developed for the verification and validation of quantitative system aspects; we mention timed automata [3], probabilistic CTL [17], hybrid bisimilarity [18].

However, the analysis within these frameworks is still boolean: either a timed automaton satisfies a property or not; two hybrid automata are either bisimilar or they are not. A boolean approach to quantitative system analysis suffers from the drawback of being fragile: small perturbations in the values within the system description may lead to opposite truth values for the satisfaction of a property. This is problematic, since the system values are usually only known approximately, because they are often obtained by measurement, learning or educated guesses.

To circumvent this problem, quantitative methods for quantitative system analysis have been proposed [5,6,8,13]. These approaches are based on quantitative logics and quantitative system relations: whereas boolean logics indicate whether

©2008 Published by Elsevier Science B. V.
a property holds for a system or not, quantitative logics express to what extent a property holds for a system; whereas (boolean) bisimulations indicate whether or not two systems are equivalent, quantitative system relations (or distances) measure how similar two systems are. More specifically, [6] introduces QC\textit{tl}, a quantitative analog of C\textit{tl}, together with model checking procedures for it. [5] considers QL\textit{tl}, a quantitative analog of L\textit{tl}, and a quantitative $\mu$-calculus and characterizes these by quantitative versions of trace equivalence and bisimilarity, respectively.

This paper continues the quest for quantitative verification and provides model checking procedures for QL\textit{tl} over quantitative transition systems (QTSs) and quantitative Markov chains (QMCs). QTSs and QMCs are resp. Kripke structures and Markov Chains whose atomic propositions have values in $[0,1]$, rather than in $\{0,1\}$. We also extend QL\textit{tl} with a quantitative until operator, which is not present in [5].

Our model checking procedure generalizes the classical L\textit{tl} model checking algorithms and constructs a Büchi automaton $A_{\varphi}$ for each QL\textit{tl} formula $\varphi$.

As the classical L\textit{tl} model checking algorithm, our procedure constructs a Büchi automaton $A_{\varphi}$ for each QL\textit{tl} formula $\varphi$. Our construction of $A_{\varphi}$ bears many similarities to the L\textit{tl} case: Recall that, for L\textit{tl}, each state $q$ in $A_{\varphi}$ is a subset of the closure of $\varphi$, which, roughly speaking, contains all subformulas of $\varphi$. In our case, each state $q$ in $A_{\varphi}$ assigns a value $\gamma(\psi)$ (from a finite subset of $[0,1]$) to each formula $\psi$ in the closure of $\varphi$. The correctness of this construction heavily relies on the fact that the Büchi conditions for the boolean until operator immediately generalize to the quantitative case: for each formula $\psi_1 U \psi_2$ in the closure of $\varphi$, we require that a trace accepted by $A_{\varphi}$ hits infinitely many times a state where $\gamma(\psi_1 U \psi_2) = \gamma(\psi_2)$. On the other hand, there are also several striking differences in the model checking algorithms for L\textit{tl} and QL\textit{tl}. As an example, when model checking $\varphi$ over a QTS $S$, we consider $S \times A_{\varphi}$, rather than $S \times A_{\neg\varphi}$, hence avoiding any complementation operation. Indeed, contrary to the L\textit{tl} case, we do not test whether all the executions satisfy the property, but rather compute the minimal value for which the property is satisfied. The later can only be done by combining the automaton and the system, looking for the minimal among all the accepting executions.

We show that for Markov chains, the model checking problem for QL\textit{tl} reduces to the one for the L\textit{tl} case, and that it has no additional cost. One interesting aspect of our approach is that the automaton $A_{\varphi}$ we build is separated, i.e. all states accept disjoint languages. Following [11], this allows to avoid the use of Rabin automata, matching the well-known single exponential complexity bound proposed in [12].

Finally, we conclude the paper with several open problems and extensions. First, we state that the model checking procedure for QL\textit{tl} over quantitative Markov decision processes is still partially open. Indeed, as for Markov chains, we show that this problem can be reduced to the model checking problem for L\textit{tl} over Markov decision process. However, contrary to the Markov chain case, this reduction has an exponential cost. We also consider three extensions of QL\textit{tl}. First, we deal with the logic QC\textit{tl}*, which is obtained by adding path quantifiers $\exists$ and $\forall$ to QL\textit{tl}. By interpreting the path quantifiers as in [6], we obtain that, on QMCs, model checking QC\textit{tl}* is directly equivalent to model checking QL\textit{tl}, while on QTSs it can be reduced to model checking QL\textit{tl}. Further extensions include one where temporal operators are equipped with discount factors, and the another one
faella, legay, stoelinga

featuring a long-run average operator. Model checking procedures for those two extensions is open, and the paper clearly states where the difficulties are.

organization of the paper. In section 2, we briefly recall some theory on automata over infinite words. section 3 introduces qts and qmc models, section 4 presents the logic qltl, while section 5 treats our model checking algorithms. Then, we present in section 6 several extensions to the theory and in section 7 some conclusions.

2 background on infinite-word automata

we suppose the reader familiar with the theory of finite-word automata. We recall basic notions and definitions concerning infinite words and infinite-word automata. An infinite word (or ω-word) w over an alphabet Σ is a mapping w : N→Σ. The set of infinite words over Σ is denoted Σω.

we consider sets of infinite words that can be represented by automata. Formally, an infinite-word automaton is a tuple A = (Σ, Q, Q0, ρ, F), where Σ is a finite alphabet, Q is a finite set of states, Q0 is a set of initial states, ρ : Q × Σ → 2Q is a nondeterministic transition function, and F is an acceptance condition. The automaton A is said to be deterministic iff |ρ(q,a)| = 1 for each a ∈ Σ, q ∈ Q. A run π of A on an infinite word w is a mapping π : N→Q, with π(0) ∈ Q0, and for all i ≥ 0, π(i + 1) ∈ ρ(π(i), w(i)).

acceptance of a run π is defined in terms of the set of states that occur infinitely often in π. This set is denoted by inf(π). We consider the following types of acceptance conditions.

• a Büchi condition is a set F ⊆ Q of accepting states. A set T ⊆ Q is accepting for the Büchi condition if T ∩ F ≠ ∅.
• a generalized Büchi condition is a subset F of 2Q. A set T ⊆ Q is accepting for the generalized Büchi condition iff for each Fi ∈ F, T ∩ Fi ≠ ∅.
• a Rabin condition is a subset F of 2Q×2Q, i.e., it is a collection of pairs of sets of states, written [(L1, U1)...(Ln, Un)]. A set T ⊆ Q is accepting for the Rabin condition if T ∩ Li ≠ ∅ and T ∩ Ui = ∅ for some i.

a Büchi (resp. generalized Büchi, Rabin) automaton A is an automaton on infinite words with a Büchi (resp. a generalized Büchi, a Rabin) acceptance condition. A word w is accepted by A if there exists a run π on w such that the set inf(π) is accepting with respect to the Büchi (resp. generalized Büchi, Rabin) condition. The set of infinite words accepted by A is called the language of A and is denoted by L(A). We denote by LQx(A) the language accepted by A when considering Qx to be the set of initial states. The automaton A is separated iff for each q, q' ∈ Q such that q ≠ q', it holds L_{(q)}(A) ∩ L_{(q')} (A) = ∅.

Büchi condition is a special case of both generalized Büchi and Rabin conditions. Hence, Büchi automata are not more expressive than generalized Büchi and Rabin automata. The opposite direction also holds.

It is well known that finite-word automata are closed under determinization. When working with infinite-word automata, this closure property may not hold.
As an example, Büchi and generalized Büchi automata are not closed under determinization. On the other hand, Rabin automata are closed under determinization. The following theorem is known to have a significant impact in many automata-based model checking algorithms.

**Theorem 2.1** Given a Büchi automaton $A$, there is a deterministic Rabin automaton $A'$ such that $L(A) = L(A')$.

Theorem 2.1 was first stated in [21], where a doubly exponential construction was provided. This was improved in [23], where a singly exponential, with an almost linear exponent, construction was provided (if $A$ has $n$ states, then $A'$ has $2^{O(n \log n)}$ states and $O(n)$ pairs in its acceptance condition).

# 3 Models

## 3.1 Basic Definitions

We introduce some notations that will be used throughout the rest of the paper. For two real numbers $u_1$ and $u_2$, we write $u_1 \uplus u_2$ for $\max\{u_1, u_2\}$, and $u_1 \ominus u_2$ for $\min\{u_1, u_2\}$. Given a set $E$ and a sequence $\pi = e_0 e_1 e_2 \cdots \in E^\omega$, we write $\pi_i$ for the $i$-th element $e_i$ of $\pi$, and we write $\pi^i = e_i e_{i+1} e_{i+2} \cdots$ for the (infinite) suffix of $\pi$ starting from $\pi_i$. Let $\Sigma$ be a finite set and $X \subseteq [0, 1]$, we denote by $\text{vals}(\Sigma, X)$ the set of all functions from $\Sigma$ to $X$. All elements of $\text{vals}(\Sigma, X)$ are called $\Sigma$-valuations. We denote by $\text{trac}(\Sigma, X)$ the set of infinite sequences of valuations from $\text{vals}(\Sigma, X)$.

All elements of $\text{trac}(\Sigma, X)$ are called $\Sigma$-traces.

## 3.2 Quantitative Transition Systems

A quantitative transition system (QTS for short) $S = (\Sigma, S, \delta, [\cdot])$ consists of a set $\Sigma$ of atomic propositions, a finite set $S$ of states, a transition relation $\delta \subseteq S \times S$, which assigns to each state a nonempty set of successor states, and a function $[\cdot] : S \to (\Sigma \to [0, 1])$ which assigns to each state $s \in S$ and proposition $r \in \Sigma$ a real value $[s](r)$. The size of $S$ is given by its number of transitions. We denote by $\mathcal{V}(S)$ the set of all values taken by the atomic propositions in any state of $S$. In addition, for each such value $x$, $1 - x$ also belongs to $\mathcal{V}(S)$. Formally, $\mathcal{V}(S) = \{[s](r) \mid s \in S, r \in \Sigma\} \cup \{1 - [s](r) \mid s \in S, r \in \Sigma\}$.

A path in $S$ is an infinite sequence $\pi = s_0 s_1 s_2 \cdots$ of states such that $(s_i, s_{i+1}) \in \delta$ for all $i \in \mathbb{N}$. Given a state $s$, we write $\text{pts}(s)$ for the set of all paths starting in $s$. Every path $\pi$ in $S$ induces the $\Sigma$-trace $[\pi] = [\pi_0][\pi_1][\pi_2] \cdots$. With an abuse of notation, we write $\text{trac}(s) = \{[\pi] \mid \pi \in \text{pts}(s)\}$ for the set of $\Sigma$-traces from $s \in S$. Notice that $\text{trac}(s) \subseteq \text{trac}(\Sigma, [0, 1])$.

## 3.3 Quantitative Markov Chains

Given a finite set $S$, a probability distribution on $S$ is a function $\mu : S \to [0, 1]$ such that $\sum_{s \in S} \mu(s) = 1$. We denote by $\mathcal{D}(S)$ the set of all probability distributions on $S$. A quantitative Markov chain (QMC for short) $S = (\Sigma, S, \Delta, [\cdot])$ consists of a set $\Sigma$ of atomic propositions, a finite set $S$ of states, a transition relation $\Delta : S \to \mathcal{D}(S)$, and
a function \([\cdot] : S \rightarrow (\Sigma \rightarrow [0, 1])\). The size of \(S\) is given by \(|S|^2\). A QMC \((\Sigma, S, \Delta, [\cdot])\) induces a QTS \((\Sigma, S, \delta, [\cdot])\), where \(\delta = \{(s, t) \in S^2 \mid \Delta(s)(t) > 0\}\). Definitions for paths and traces in a QMC are identical to those for the corresponding QTS.

A quantitative Markov chain together with an initial state \(s\) gives rise to a probability space \((\text{trac}(s), \mathcal{B}, \Pr_s)\), where \(\mathcal{B}\) is the set of measurable subsets of \(\text{trac}(s)\), and \(\Pr_s\) is the uniquely induced probability measure (see [10] for an introduction). Given a random variable \(X\) over this probability space, we denote its expected value by \(E_s[X]\).

When discussing the complexity of algorithms taking a QMC as input, we assume that transition probabilities are encoded as fixed-precision numbers, and therefore that arithmetic operations and comparisons take constant time.

## 4 Quantitative LTL

In this section we introduce Quantitative Linear Temporal Logic (QLTL for short), a quantitative version of the Linear Temporal Logic (LTL for short) introduced in [22].

### 4.1 Syntax

Let \(\Sigma\) be a set of atomic propositions. The QLTL formulas over \(\Sigma\) are generated by the following grammar:

\[
\varphi ::=} r \mid t \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \neg \varphi \mid \varphi U \varphi \mid \varphi U \neg \varphi \mid \varphi \Diamond \varphi \mid \varphi \Box \varphi
\]

where \(r \in \Sigma\). The operators \(U, \Diamond, \Box\) are the temporal operators. The syntax of QLTL is therefore the same as the one of LTL.

### 4.2 Semantics

Here \(r \in \Sigma\) is an atomic proposition. A QLTL formula \(\varphi\) over \(\Sigma\) assigns a value \([\varphi](\sigma)\) to each \(\Sigma\)-trace \(\sigma\) as follows.

\[
\begin{align*}
[r](\sigma) &= \sigma_0(r) \\
[\neg \varphi](\sigma) &= 1 - [\varphi](\sigma) \\
[\varphi U \psi](\sigma) &= \sup_{i \geq 0} ([\varphi](\sigma^i) \cap [\psi](\sigma^i)) \\
[\varphi U \neg \psi](\sigma) &= \inf_{i \geq 0} ([\varphi](\sigma^i) \cup [\neg \psi](\sigma^i)) \\
[\varphi \Diamond \psi](\sigma) &= \sup_{i \geq 0} ([\varphi](\sigma^0) \cap \cdots \cap [\varphi](\sigma^{i-1}) \cap [\psi](\sigma^i)) \\
[\varphi \Box \psi](\sigma) &= \inf_{i \geq 0} ([\varphi](\sigma^0) \cup \cdots \cup [\varphi](\sigma^{i-1}) \cup [\psi](\sigma^i)).
\end{align*}
\]

The semantics of QLTL is a proper extension of the one of LTL, in the following sense. If the value of all atomic propositions at all positions of a trace is either 0 or 1 (i.e., if the trace Boolean), then the value of a QLTL formula \(\varphi\) on such trace is the same of the value of \(\varphi\) on that trace, if \(\varphi\) is interpreted as an LTL formula, 0 is interpreted as false and 1 as true.
We observe that the following classical equivalences hold:

\[
\[\square \varphi\](\sigma) = [\varphi \land U \varphi](\sigma) \quad \quad [\diamond \varphi](\sigma) = [\top \lor U \varphi](\sigma).
\]

In the logic we have defined, negation can be applied to any subformula. However, any QLTL formula is equivalent to a formula where negation is only applied to atomic propositions, according to the following equivalences.

\[
[\neg (\varphi_1 U \varphi_2)](\sigma) = [(\neg \varphi_1) \land (\neg \varphi_2)](\sigma)
\]

\[
[\neg (\varphi_1 U \varphi_2)](\sigma) = [(\neg \varphi_1) \lor (\neg \varphi_2)](\sigma)
\]

\[
[\neg (\varphi_1 \land \varphi_2)](\sigma) = [\neg \varphi_1 \lor \neg \varphi_2](\sigma)
\]

As a consequence, in the following we only consider formulas containing connectives \(\land, \lor, \neg, \circ, U, \) and \(U\), and where negation is only applied to atomic propositions.

**Evaluation for quantitative transition systems.** A QLTL formula \(\varphi\) assigns a real value \([\varphi](s) \in [0, 1]\) to each state \(s\) of a given QTS, according to the rule \([\varphi](s) = \sup \{[\varphi](\sigma) | \sigma \in \text{trac}(s)\}\).

**Evaluation for quantitative Markov chains.** Given a QMC \(S\), a state \(s\), and a QLTL formula \(\varphi\), the function \([\varphi]\), which assigns a real value to each \(\Sigma\)-trace, is a random variable over the probability space \((\text{trac}(s), B, \Pr_s)\). Accordingly, we define the value of \(\varphi\) on state \(s\) to be \([\varphi](s) = E_s[\varphi]\).

5 Evaluating QLTL

In this section, we extend the automata-based technique by [27,28] to determine the valuation of a QLTL formula on a QTS or a QMC. First, we prove that if along a trace all atomic propositions only take a finite number of different values, any QLTL formula assigns to that trace either one of the values occurring in the trace, or \(1 - x\), where \(x\) is a value occurring in the trace. As a corollary, when evaluated on a QTS \(S\), a QLTL formula can only assume value in \(\mathcal{V}(S)\).

**Theorem 5.1** Let \(\mathcal{V}\) be a finite subset of \([0, 1]\) and let \(\sigma \in \text{trac}(\Sigma, \mathcal{V})\). Then, for all QLTL formulas \(\varphi\), we have \([\varphi](\sigma) \in \mathcal{V} \cup \{1 - x | x \in \mathcal{V}\}\).

**Proof.** We proceed by structural induction on \(\varphi\). The thesis is obviously true when \(\varphi\) is an atomic proposition. It is easy to check that all temporal operators do not enrich the range of possible values for the formulas.

**Corollary 5.2** Given a QTS \(S\), a state \(s \in S\), and a QLTL formula \(\varphi\), we have \([\varphi](s) \in \mathcal{V}(S)\).

Next, we consider the two following definitions.

**Definition 5.3** The closure of a QLTL formula \(\varphi\) is the smallest set \(\text{clos}(\varphi)\) of QLTL
formulas such that:

\[
\varphi \in \text{clos}(\varphi) \quad \psi_1 \lor \psi_2 \in \text{clos}(\varphi) \implies \psi_1, \psi_2 \in \text{clos}(\varphi) \\
\psi_1 \land \psi_2 \in \text{clos}(\varphi) \implies \psi_1, \psi_2 \in \text{clos}(\varphi) \\
\Diamond \psi_1 \in \text{clos}(\varphi) \implies \psi_1 \in \text{clos}(\varphi) \\
\psi_1 \mathcal{U} \psi_2 \in \text{clos}(\varphi) \implies \psi_1, \psi_2 \in \text{clos}(\varphi) \\
\psi_1 \mathcal{U} \psi_2 \in \text{clos}(\varphi) \implies \psi_1, \psi_2 \in \text{clos}(\varphi).
\]

We denote by \(|\varphi|\) the number of temporal operators, boolean connectives and propositions found in the formula \(\varphi\). Notice that \(|\text{clos}(\varphi)| = O(|\varphi|)\).

**Definition 5.4** A closure-valuation for a \(\text{QLtl}\) formula \(\varphi\) is a function \(v : \text{clos}(\varphi) \rightarrow [0,1]\). A closure-valuation is consistent if the following conditions hold.

(i) If \(t \in \text{clos}(\varphi)\), then \(v(t) = 1\).
(ii) If \(r \in \text{clos}(\varphi)\), then \(v(r) = 0\).
(iii) If \(\psi_1 \lor \psi_2 \in \text{clos}(\varphi)\), then \(v(\psi_1 \lor \psi_2) = v(\psi_1) \lor v(\psi_2)\).
(iv) If \(\psi_1 \land \psi_2 \in \text{clos}(\varphi)\), then \(v(\psi_1 \land \psi_2) = v(\psi_1) \land v(\psi_2)\).
(v) If both \(r\) and \(\neg r\) belong to \(\text{clos}(\varphi)\), then \(v(\neg r) = 1 - v(r)\).

A closure-trace is an infinite sequence of consistent closure-valuations.

To determine the value of a \(\text{QLtl}\) formula on a \(\Sigma\)-trace one can proceed by building a closure-trace in a way that is compatible with \(\text{QLtl}\) semantics. Consider a closure-trace \(\gamma\) for a formula \(\varphi\) defined over a set of atomic propositions \(\Sigma\). For a \(\Sigma\)-trace \(\sigma\), we say that \(\gamma\) is valid for \(\sigma\) if it satisfies the following rules for each \(i \geq 0\) (adapted from [28]):

(i) For each \(r \in \Sigma\), if \(r \in \text{clos}(\varphi)\) then \(\gamma_i(r) = \sigma_i(r)\), and if \(\neg r \in \text{clos}(\varphi)\) then \(\gamma_i(\neg r) = 1 - \sigma_i(r)\).
(ii) If \(\gamma_i(\Diamond \psi_1) = u\) then \(\gamma_{i+1}(\psi_1) = u\).

For the \(\mathcal{U}\) and \(\mathcal{U}\) operators, the semantic rules refer to a possibly infinite set of points of the sequence. The solution is first to notice that the following identities hold for each \(i \geq 0\):

\[
[\psi_1 \mathcal{U} \psi_2](\sigma^i) = [\psi_2](\sigma^i) \cup ([\psi_1](\sigma^i) \cap [\Diamond(\psi_1 \mathcal{U} \psi_2)](\sigma^i)) \\
[\psi_1 \mathcal{U} \psi_2](\sigma^i) = [\psi_2](\sigma^i) \cap ([\psi_1](\sigma^i) \cup [\Diamond(\psi_1 \mathcal{U} \psi_2)](\sigma^i)).
\]

These identities suggest the following labeling rules for each \(i \geq 0\):

(iii) If \(\gamma_i(\psi_1 \mathcal{U} \psi_2) = u\), then \(u = \gamma_i(\psi_2) \cup (\gamma_i(\psi_1) \cap \gamma_{i+1}(\psi_1 \mathcal{U} \psi_2))\).
(iv) If \(\gamma_i(\psi_1 \mathcal{U} \psi_2) = u\), then \(u = \gamma_i(\psi_2) \cap (\gamma_i(\psi_1) \cup \gamma_{i+1}(\psi_1 \mathcal{U} \psi_2))\).

However, as it is illustrated by the following example, those conditions are not sufficient for the closure-trace to be valid.

**Example 5.5** Consider the QLtl formula \(p \mathcal{U} q\) with \(p, q \in \Sigma\). We have \(\text{clos}(\varphi) = \{p \mathcal{U} q, p, q\}\). Consider now a closure-trace that constantly assigns values 0.6, 0.7,
and 0.3 to \( pUq, p, \) and \( q, \) respectively. This trace is valid for any \( \Sigma \)-trace that always assigns the value 0.7 to \( p \) and 0.3 to \( q \). However, the evaluation of \( pUq \) on such a trace would be 0.3, and thus not 0.6 as it is suggested by the closure-trace.

The problem in the example above is that when only considering rules (iii) and (iv), the evaluation of \( pUq \) can always be postponed to the next element in the sequence. The solution is to observe that since the systems on which QLTL-formula are evaluated are finite-state systems, one can restrict ourselves to a finite subset of \([0, 1]\). In this setting, we obtain the following result.

**Theorem 5.6** Consider a QLTL formula of the form \( \varphi_1 U \varphi_2 \) (resp. \( \varphi_1 U \varphi_2 \)). Let \( \mathcal{V} \) be a finite subset of \([0, 1]\) and let \( \sigma \in trac(\Sigma, \mathcal{V}) \). For all \( i \geq 0 \) there exists \( j \geq i \) such that \( [\varphi_1 U \varphi_2](\sigma_j) = [\varphi_2](\sigma_j) \) (resp. \( [\varphi_1 U \varphi_2](\sigma_j) = [\varphi_2](\sigma_j) \)).

**Proof.** Consider the \( \mathcal{U} \) case. The proof is a direct consequence of the semantic and the fact that \( \mathcal{V} \) is finite. By contradiction, one could extract an infinite sequence where the evaluation of \( \varphi_2 \) is strictly increasing. The \( \tilde{\mathcal{U}} \) case is proved similarly. \( \square \)

As a consequence of Theorem 5.6, we add the following labeling rules, which only have sense when considering \( \mathcal{V} \) to be a finite subset of \([0, 1] \):

(v) For each \( i \geq 0 \), there exists \( j \geq i \) such that \( \gamma_j(\psi_1 U \psi_2) = \gamma_j(\psi_2) \).

(vi) For each \( i \geq 0 \), there exists \( j \geq i \) such that \( \gamma_j(\psi_1 \tilde{U} \psi_2) = \gamma_j(\psi_2) \).

**Theorem 5.7** Consider a QLTL formula \( \varphi \), a finite set \( \mathcal{V} \subseteq [0, 1] \) and a \( \Sigma \)-trace \( \sigma \in trac(\Sigma, \mathcal{V}) \). We have that \( [\varphi](\sigma) = u \) if and only if there exists a valid closure-trace \( \gamma \) for \( \sigma \) such that \( \gamma_0(\varphi) = u \).

Given a QLTL formula \( \varphi \), we now build a generalized Büchi automaton that describes a possibly infinite set of \( \Sigma \)-traces and whose states are consistent closure-valuations of \( \varphi \). More precisely, the automaton is built in such a way that for each formula \( \psi_1 \) in the closure of \( \varphi \), for each state \( q \), and for each accepting \( \Sigma \)-trace \( \sigma \) from \( s \), the valuation \( [\psi_1](\sigma) \) is given by \( q(\psi_1) \).

**Definition 5.8** Let \( \Sigma \) be a set of atomic propositions and let \( \mathcal{V} \) be a finite subset of \([0, 1]\) such that, for all \( x \in \mathcal{V}, 1 - x \in \mathcal{V} \). We define the QLTL-automaton for \( \varphi \) and \( \mathcal{V} \) as the tuple \( A^V_\varphi = (vals(\Sigma, \mathcal{V}), Q, Q_0, \rho, F) \), where:

- The alphabet of the automaton is \( vals(\Sigma, \mathcal{V}) \).
- The set of states \( Q \) is the set of closure-valuations in \( vals(clos(\varphi), \mathcal{V}) \) which are consistent.
- We choose \( Q_0 = Q \).
- The transition function is such that for each \( q, q' \in Q \) and \( a \in vals(\Sigma, \mathcal{V}) \), we have \( q' \in \rho(q, a) \) if
  (i) For all \( r \in \Sigma \), if \( r \in clos(\varphi) \) (resp. \( \neg r \in clos(\varphi) \)), then \( q(r) = a(r) \) (resp. \( q(\neg r) = 1 - a(\neg r) \)).
  (ii) If \( \circ \psi_1 \in clos(\varphi) \), then \( q(\circ \psi_1) = q'(\psi_1) \).
  (iii) If \( \psi_1 U \psi_2 \in clos(\varphi) \), then \( q(\psi_1 U \psi_2) = q(\psi_2) \cup (q(\psi_1) \cap q'(\psi_1 U \psi_2)) \).
  (iv) If \( \psi_1 \tilde{U} \psi_2 \in clos(\varphi) \), then \( q(\psi_1 \tilde{U} \psi_2) = q(\psi_2) \cap (q(\psi_1) \cup q'(\psi_1 \tilde{U} \psi_2)) \).
• If \( \text{clos}(\varphi) \) contains no formula with \( \mathcal{U} \) and \( \bar{\mathcal{U}} \) operators, then \( F = \{ Q \} \). Otherwise, for each formula of the form \( \psi_1 \mathcal{U} \psi_2 \) (resp. \( \psi_1 \bar{\mathcal{U}} \psi_2 \)) in \( \text{clos}(\varphi) \), \( F \) contains the set \( \text{clos}_v \mathcal{U} \psi_2 \) (resp. \( \text{clos}_v \bar{\mathcal{U}} \psi_2 \)), where \( q \in \text{clos}_v \mathcal{U} \psi_2 \) iff \( q(\psi_1 \mathcal{U} \psi_2) = q(\psi_2) \) (resp. \( q(\psi_1 \bar{\mathcal{U}} \psi_2) = q(\psi_2) \)).

Observe that the number of states of \( A^\varphi_{\psi} \) is bounded by \( |\mathcal{V}|^{\text{clos}(\varphi)} \). In practice the bound is not reach since one only considers consistent closures. The set of initial states is defined arbitrarily and will be discussed in the next section. The following theorem is a key result, stating the correctness of construction \( A^\varphi_{\psi} \).

**Theorem 5.9** Let \( \varphi \) be a QLTL formula and let \( \mathcal{V} \) be a finite subset of \([0,1]\) such that if \( x \in \mathcal{V} \), then also \( 1-x \in \mathcal{V} \). Then we have for all \( v \in \mathcal{V} \) and all \( \psi \) in the closure of \( \varphi \) that

\[
\bigcup_{q \in Q, q(\varphi) = v} L_q(A^\varphi_{\psi}) = \{ \nu \in \text{vals}(\Sigma, \mathcal{V}) \mid [\psi](\sigma) = v \}.
\]

**Proof.** By induction on the structure of \( \psi \), using Theorem 5.7. \( \square \)

The automaton \( A^\varphi_{\psi} \) also satisfies the following property, which will be of particular interest for the results that will be presented in Section 5.2.

**Theorem 5.10** The automaton \( A^\varphi_{\psi} \) is separated.

**Proof.** For each \( q, q' \in Q \) with \( q \neq q' \), there exists \( \varphi_1 \in \text{clos}(\varphi) \) such that \( q(\varphi_1) \neq q'(\varphi_1) \). Since it is not possible that a \( \Sigma \)-trace assigns two different values to the same formula \( \varphi_1 \), we have \( L_q(A^\varphi_{\psi}) \cap L_{q'}(A^\varphi_{\psi}) = \emptyset \). \( \square \)

### 5.1 Evaluating QLTL on Quantitative Transition Systems

Consider a quantitative transition system \( S = (\Sigma, S, \delta, [\cdot]) \) and a QLTL formula \( \varphi \). We aim at computing \( [\varphi](s) \) for a state \( s \in S \). We first propose the following definition

**Definition 5.11** Consider a QTS \( S = (\Sigma, S, \delta, [\cdot]) \) and a QLTL formula \( \varphi \). Let \( A_{\varphi} = (\text{vals}(\Sigma, \mathcal{V}(S)), Q, Q_0, \rho, F) \) be the QLTL-automaton for \( \varphi \) and \( \mathcal{V}(S) \). For a state \( s \in S \), the \( \bar{s} \)-product of \( S \) and \( A_{\varphi} \), denoted \( S \times A_{\varphi} \), is the automaton \((\emptyset, Q', \rho', F')\), where:

- The alphabet contains only the symbol \( \emptyset \).
- The set of states \( Q' \) contains all pairs \((s, q) \in S \times Q\) which are synchronized w.r.t. the value of the atomic propositions. Formally, for all \( r \in \text{clos}(\varphi) \), \([s](r) = q(r)\).
- The set of initial states is given by \( Q'_0 = ([s] \times Q_0) \cap Q' \).
- The set of final states is given by \( F' = (S \times F) \cap Q' \).
- We have \((s', q') \in \rho'((s, q), \emptyset)\) iff \((s, s') \in \delta\).

Our approach to computing \( [\varphi](s) \) consists in the following three steps:

(i) We first build the \( \bar{s} \)-product \( S \times A_{\varphi} \) between the system \( S \) and the QLTL-automaton for \( \varphi \) and \( \mathcal{V}(S) \).
(ii) We then compute the set of states $Q'' = \{ (\bar{s}, q) \in Q'_0 \mid L((\bar{s}, q)) (S \times A_\varphi) \neq \emptyset \}$.

(iii) Finally, $[\varphi](\bar{s}) = \min_{(\bar{s}, q) \in Q''} q(\varphi)$.

As far as the complexity of the above procedure is concerned, it is easy to see that step (ii) dominates the others. Such step consists in determining the set $Q''$ of states of the product automaton which, used as initial states, give rise to a non-empty language. The classical algorithm for the emptiness of a generalized Büchi automaton can be easily adapted to compute the set $Q''$ in time linear in the size of the product (precisely, in the number of edges in the product). We thus obtain the following theorem which states that the model checking procedure for $\text{QLTL}$ is not more expensive than the one for model checking $\text{LTL}$.

**Theorem 5.12** Given a $\text{QLTL}$-formula $\varphi$, a QTS $S = (\Sigma, S, \delta, [\cdot])$, and a state $s \in S$, the value $[\varphi](s)$ can be computed in time $O(|\delta| \cdot |V(S)|^{\lfloor \text{close}(\varphi) \rfloor})$.

Notice that, unlike the $\text{LTL}$ case, our evaluation procedure does not need to complement a Büchi automaton or a $\text{QLTL}$ formula.

### 5.2 Evaluating $\text{QLTL}$ on Quantitative Markov Chains

In this section, we consider the model checking problem for $\text{QLTL}$ over quantitative Markov chains. We will show that this problem can be reduced to the model checking problem for $\text{LTL}$ over Markov chains.

Consider a QMC $S = (\Sigma, S, \Delta, [\cdot])$ and a $\text{QLTL}$ formula $\varphi$ on $\Sigma$. We aim at computing $E_s([\varphi])$ for a state $s \in S$. Assuming that $V(S) = \{b_1, b_2, \ldots, b_n\}$, recall that we denote by $\Pr_s([\varphi] = b_i)$ the probability for the value of the random variable $[\varphi]$ to be $b_i$ on the probability space generated by the traces starting at $s$. We have

$$[\varphi](s) = E_s([\varphi]) = \sum_{i=1}^{n} b_i \cdot \Pr_s([\varphi] = b_i).$$

Consequently, to compute $[\varphi](s)$, it is sufficient to compute for each value $b_i \in V(S)$ the probability for the value of the random variable to be $b_i$. More precisely, given the set of $\Sigma$-traces $T^{b_i} = \{ \sigma \in \text{trac}(S) \mid [\varphi](\sigma) = b_i \}$ and the probability space $(\text{trac}(s), B, \Pr_s)$ given by $S$ and $s$, we aim at computing $\Pr_s(T^{b_i})$. For this, we recall the following theorem (see [10] for a proof).

**Theorem 5.13** Consider a QMC $S = (\Sigma, S, \Delta, [\cdot])$, a state $s \in S$, and the probability space $(\text{pts}(s), B, \Pr_s)$ given by $S$ and $s$. Let $T$ be a set of $\Sigma$-traces. If $T$ can be represented by a deterministic Rabin automaton with $n$ states, then one can compute $\Pr_s(T)$ in time polynomial in $|S| \cdot n$.

We thus need to provide a deterministic Rabin automaton $A_{\varphi = b_i}$ accepting $T^{b_i}$, for each $b_i$. Working with a deterministic Rabin automaton is needed not to break the deterministic behavior of Markov chain (see [10]). The automaton $A_{\varphi = b_i}$ can easily be obtained from the automaton $A_\varphi$. Indeed, it suffices to remove from the set of initial states of $A_\varphi$ all the states that do not assign the value $b_i$ to $\varphi$. We obtain a generalized Büchi automaton, which can be turned into a deterministic Rabin one whose size is exponentially larger (see Theorem 2.1).
The result above involves a double exponential, which is due to the fact that we build a generalized Büchi automaton for the formula (whose size is exponential in the size of the formula), and then turn it into a deterministic Rabin one (whose size is again exponential in the size of the Büchi). However, in [11], it is showed that one can avoid the exponential blow-up needed to compute the deterministic Rabin automaton, when the generalized Büchi automaton representing the formula is separated. Observe that since the automaton $A_\phi$ is separated, any automaton $A_{\phi=b_i}$ will also be separated. We can thus use the result from [11] to avoid one exponential blow-up. The resulting algorithm is polynomial in the size of the QMC and singly exponential w.r.t. the formula.

Remark 5.14 The result in [11] additionally requires automata to be unambiguous. An automaton is unambiguous if two transitions that start in the same state and have the same label reach different destinations. This property is satisfied by our automata by definition. In conclusion, the automaton $A_\phi$ is separated and unambiguous. Moreover, this property does not depend on the set of initial states.

6 Extensions and Open Problems

This section discusses several extensions of QLTL model checking. First, we describe how our QLTL model checking algorithm can be extended to an algorithm for QC\textit{C}TL*. Then, we present a partial solution to the model checking problem for a discounted version of QLTL. Finally, model checking the $\Delta$ operator and quantitative Markov decision processes remain completely open.

6.1 From QLTL to QC\textit{C}TL*

Having considered the branching logic DC\textit{C}TL in [6] and the linear logic QLTL in this paper, it is natural to consider logic QC\textit{C}TL*, which extends QLTL with path quantifiers $\exists$ and $\forall$. The syntax of QC\textit{C}TL* is the same as the one of C\textit{C}TL*. The semantics of a QC\textit{C}TL* formula is defined with respect to the system on which it is evaluated. Consider a QC\textit{C}TL* formula $\phi$.

- If the formula is evaluated for a state $s$ of a quantitative transition system $S$, then the operators $\forall$ and $\exists$ represent the inf-evaluation-over-all and the sup-evaluation-over-all traces, respectively. Formally, $[\forall \phi](s) = \inf \{ [\phi](\sigma) | \sigma \in \text{trac}(s) \}$, and $[\exists \phi](s) = \sup \{ [\phi](\sigma) | \sigma \in \text{trac}(s) \}$. Observe also that $[\exists \phi](s) = 1 - [\forall \neg \phi](s)$.
- When considering quantitative Markov chains, following [6], we interpret both $\forall$ and $\exists$ as the expected value operator. Therefore, on QMCs, QC\textit{C}TL* essentially coincides with QLTL.

Evaluating QC\textit{C}TL* formulas with only one path quantifier is immediate. Indeed, the automata-based algorithm presented in Section 5.1 allows us to immediately evaluate formulas of the form $\exists \phi$. Observing that $[\exists \phi](s) = 1 - [\forall \neg \phi](s)$, we get the result. When considering formulas with several path quantifiers, one recursively replaces each quantified subformula with a new atomic proposition that represents its value (using again the automata-based algorithm). We thus have the following complexity result.
Theorem 6.1 Consider a QTS $S = (\Sigma, S, \delta, \cdot)$, a state $s \in S$, and a $\text{QCTL}^*$-formula $\varphi$. The value $\llbracket \varphi \rrbracket(s)$ can be computed in time $O(|\varphi| \cdot |S| \cdot |\mathcal{V}(S)|^{\text{clos}(\varphi)})$.

Recall from Theorem 5 of [6] that a formula $\varphi$ from the logic $\text{DCTL}$ can be evaluated on a QTS $S$ in time $O(|S|^2 \cdot |\varphi|)$. It follows that $\text{QLTL}$ formulas which do not contain nesting of linear operators can be evaluated in the same time. As a side note, it should be noted that $\text{DCTL}$ does not feature an until operator. However, it is our belief that its addition would not increase the complexity of model checking the logic. We therefore observe, as expected, that ad-hoc algorithms for dealing directly with $\square$ and $\Diamond$ operators are definitely more efficient than the present automata-based algorithms, which on the other hand is capable of treating arbitrary nesting of temporal operators.

6.2 Discounting

The logics in [6] use discounting, meaning that values in the near future weigh heavier than values in the far future. Given a discount factor $\alpha \in [0, 1]$, discounted versions $\circ_\alpha$, $\diamond_\alpha$, and $\square_\alpha$ of the next, eventually, and always operator are defined below. There is a second next operator $\hat{\diamond}_\alpha$, which is the dual of $\circ_\alpha$

$$\llbracket \circ_\alpha \varphi \rrbracket(\sigma) = \alpha \llbracket \varphi \rrbracket(\sigma^1)$$
$$\llbracket \hat{\diamond}_\alpha \varphi \rrbracket(\sigma) = 1 - \alpha + \alpha \llbracket \varphi \rrbracket(\sigma^1)$$
$$\llbracket \diamond_\alpha \varphi \rrbracket(\sigma) = \sup_{i \geq 0} \alpha^i \llbracket \varphi \rrbracket(\sigma^i)$$
$$\llbracket \hat{\circ}_\alpha \varphi \rrbracket(\sigma) = \inf_{i \geq 0} 1 - \alpha^i (1 - \llbracket \varphi \rrbracket(\sigma^i))$$

Just as for the next operator, one should consider two discounted variants $\mathcal{U}_\alpha$ and $\hat{\mathcal{U}}_\alpha$ of $\mathcal{U}$ (and also two for $\hat{\mathcal{U}}$)

$$\llbracket \varphi \mathcal{U}_\alpha \psi \rrbracket(\sigma) = \sup_{i \geq 0} \alpha^0 \llbracket \varphi \rrbracket(\sigma^0) \cap \alpha^1 \llbracket \psi \rrbracket(\sigma^1) \cap \ldots \cap \alpha^{i-1} \llbracket \varphi \rrbracket(\sigma^{i-1}) \cap \alpha^i \llbracket \psi \rrbracket(\sigma^i)$$
$$\llbracket \varphi \hat{\mathcal{U}}_\alpha \psi \rrbracket(\sigma) = \sup_{i \geq 0} 1 - \alpha^0 (1 - \llbracket \varphi \rrbracket(\sigma^0)) \cap 1 - \alpha^1 (1 - \llbracket \varphi \rrbracket(\sigma^1)) \cap \ldots \cap 1 - \alpha^{i-1} (1 - \llbracket \varphi \rrbracket(\sigma^{i-1})) \cap 1 - \alpha^i (1 - \llbracket \psi \rrbracket(\sigma^i))$$

For $\psi_1 \mathcal{U}_\alpha \psi_2$, we have $\llbracket \psi_1 \mathcal{U}_\alpha \psi_2 \rrbracket(\sigma^i) = \llbracket \psi_2 \rrbracket(\sigma^i) \cup (\llbracket \psi_1 \rrbracket(\sigma^i) \cap \llbracket \circ_\alpha (\psi_1 \mathcal{U}_\alpha \psi_2) \rrbracket(\sigma^i))$ and thus the following labeling rule.

$$\gamma_i(\psi_2) \sqcap (\gamma_i(\psi_1) \cap \alpha \cdot \gamma_{i+1}(\psi_1 \mathcal{U} \psi_2))$$

The other until operators can be treated similarly. It is important to realize that no Büchi conditions are needed for $\alpha < 1$: in the undiscounted case, the recursive characterization $\psi_1 \mathcal{U} \psi_2 \equiv \psi_2 \lor (\psi_1 \mathcal{U} \psi_2)$ for $\mathcal{U}$ has two fixed points and one needs the smallest. If $\alpha < 1$, then the underlying operators are contractions and have unique fixed points.

The analogon of Theorem 5.1 does not hold in the case of discounting. We do not know if, given a QTS $S$, the set of values $\mathcal{V}_\alpha(S) = \{\llbracket \varphi \rrbracket(S) \mid \varphi$ is a $\text{QLTL}$ formula with discount factor $\alpha\}$ is finite or not. However, by performing the construction of Definition 5.8 with $\mathcal{V}$ being a finite subset of $[0, 1]$, one can build an infinite-state Büchi automaton with the property of Theorem 5.9. Try to put a note about the influence of discounting in the closure, which makes it infinite. In other words, the Büchi construction works for discounting, but we cannot use it for model checking,
since it yields an infinite-state automaton. We leave it as an open problem to investigate whether \( \mathcal{V}_\alpha(S) \) is finite. If so, our construction yields a model checking algorithm for discounted QLTL. If not, alternative methods have to be investigated, e.g. based on approximation.

### 6.3 Long-run Average Operator

The branching logic DCTL of [6] also contains the path operator \( \triangle \). This operator stands for the long-run average of a quantitative proposition and is defined by:

\[
[\triangle \varphi](\sigma) = \lim_{n \to \infty} \frac{\| \varphi \|(\sigma^0) + \| \varphi \|(\sigma^1) + \cdots + \| \varphi \|(\sigma^{n-1})}{n}.
\]

Such operator does not fit well with the finite automata-based approach, since the value of a \( \triangle \) formula in general does not coincide with the value of the proposition \( p \) in any state of the system: for instance, any number in \([0,1]\) can be obtained as the long run average of a sequence whose propositional values are \{0, 0.1, 0.2, 0.3, ..., 0.9\}.

Thus, it remains open whether the \( \triangle \) operator can be evaluated on a system by automata-theoretic means.

### 6.4 Model checking QLTL over Quantitative Markov Decision Processes

A Quantitative Markov Decision Process (QMDP) is a Markov decision process (MDP) with quantitative values in the states. Thus, a QMDP can be viewed as a QMC combined with nondeterminism, i.e. each QMDP state enables one or more transitions whose target state is determined probabilistically. Thus, a QMDP \( S = (\Sigma, S, \Delta, [:]) \) contains the same ingredients as a QMC, except that the transition relation is a function \( \Delta : S \to 2^D(S) \) such that \( \Delta(s) \neq \emptyset \) for each \( s \in S \). Each QMDP induces a QTS \((\Sigma, S, \delta, [:])\), where \( \delta = \{(s, t) \in S^2 \mid \exists \mu \in \Delta(s) \land \mu(t) > 0\} \). Definitions for paths and traces in a QMDPs are identical to those for the corresponding QTS. In what follows, we assume that the number of transitions of the QMDP is \( m \).

A scheduler for \( S \) resolves the non-deterministic choices in \( S \).Schedulers can be (1) history-dependent, i.e. they may base their decisions on the history of the system and (2) randomized, i.e., they may make a probabilistic choice over the outgoing transitions in each state. More precisely, a scheduler for \( S \) in a state \( s_0 \) is a function \( \pi : pts(s_0) \to D(D(S)) \) such that if \( \pi(s_0, s_1 \ldots s_n)(\mu) > 0 \), then \( \mu \in \delta(s_n) \). A scheduler is memoryless if \( last(\rho) = last(\rho') \) implies \( \pi(\rho)(\mu) = \pi(\rho')(\mu) \) for all \( \mu \in D(S) \). A scheduler is deterministic if for each path \( \rho \) there is exactly one \( \mu \in D(S) \) with \( \pi(\rho)(\mu) > 0 \). We denote the set of all schedulers in \( s_0 \) by \( Sched(s_0) \) and set of all schedulers in \( s_0 \) that are both memoryless and deterministic by \( DSched(s_0) \).

Each scheduler \( \pi \) in state \( s \) defines a probability space over \( P^\pi_s = (\text{trac}^\pi(s), B^\pi, \text{Pr}^\pi_s) \), where \( B \) is the set of measurable subsets of \( \text{trac}(s) \), and \( \text{Pr}^\pi_s \) is the uniquely induced probability measure traces of \( S \). We denote the expected value of a random variable \( X \) over \( P^\pi_s \) by \( E^\pi_s[X] \).

We interpret QLTL over QMDPs by taking the minimum expected value over all schedulers, i.e. we set \([\varphi](s) = \inf_{\pi \in Sched(s)} E^\pi_s[[\varphi]] \).
Unfortunately, contrary to the case of Markov chains, we cannot directly extend the algorithm for \( \text{LTL} \) model checking over MDPs\([1,2,24]\): we could, for each value \( b_i \) run an \( \text{LTL} \)-inspired algorithm that finds the minimum probability \( \min_{\pi \in \text{Sched}} \Pr^\pi_s[\varphi = b_i] \) with which value \( b_i \) is attained. However, the QMDP model checking problem asks for the global minimum, i.e. \( \min_{\pi \in \text{Sched}} \sum_{i=1}^n b_i \cdot \Pr^\pi_s[\varphi = b_i] \), which cannot be found by solving the model checking problem for the \( b_i \)’s separately\(^1\). Unfortunately, we did not find a way to solve this global minimization problem based on \( \text{LTL} \) model checking.

However, we claim that, just as for \( \text{LTL} \) model checking over MDPs, the value of a \( \text{QLTL} \) formula is determined by a memoryless and deterministic scheduler, i.e. \( [\varphi](s) = \inf_{\pi \in \text{DSched}(s)} E^\pi_s[\varphi] \). Since each deterministic scheduler \( \pi \) over \( S \) induces a QMC \( S^\pi \), and there are \( O(2^n) \) different deterministic schedulers, one can model check \( \text{QLTL} \) over QMDPs by running the QMC algorithm \( O(2^n) \) times and taking the minimum of all runs, thus yielding an exponential algorithm.

In conclusion, as for Markov chains, one can still reduce the model checking problem for \( \text{QLTL} \) over quantitative MDP to the model checking problem of \( \text{LTL} \) over MDP. However, contrary to the Markov chain case, this reduction has an exponential cost since one has to consider all the schedulers. We leave the investigation of more efficient algorithms as an open problem.

7 Conclusion and Future Work

In this paper, we extended the work done in\([6]\), by presenting a quantitative linear temporal logic and showing how such logic can be model-checked (i.e., evaluated) over non-deterministic or probabilistic systems, by using a classical automata-based approach. We have provided partial solutions to the model checking problem for \( \text{QCt}^{\ast} \), and over quantitative Markov decision processes. Model checking of the long run average operator and the discounted version of \( \text{QLTL} \) is open.

Apart from the directions mentioned in Section 6, it is also worthwhile to investigate an extension of the results presented in this paper (and in\([6]\)) to continuous time or interval Markov chains. Another promising research direction consists in extending the abstract probabilistic frameworks of\([15,19]\) to quantitative logics. We could also investigate whether the alternating automata based construction of\([20]\) extends to the case of \( \text{QCTL}^{\ast} \). Finally, it would also be of interest to see whether one can reduce the size of the automata we construct following techniques similar to those proposed in\([16]\).

References


\(^1\) Indeed, since the minimal probability for each of the \( b_i \)’s could be computed with a different scheduler, the sum of all the probabilities could be greater than 1, which breaks the definition of the expected value.


