High-Quality Synthesis Against Stochastic Environments*

Shaull Almagor[†1] **and Orna Kupferman**[2]

1 **School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel**

2 **School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel**

Abstract

In the classical synthesis problem, we are given a linear temporal logic (LTL) formula ψ over sets of input and output signals, and we synthesize a transducer that realizes ψ: with every sequence of input signals, the transducer associates a sequence of output signals so that the generated computation satisfies ψ. One weakness of automated synthesis in practice is that it pays no attention to the quality of the synthesized system. Indeed, the classical setting is Boolean: a computation satisfies a specification or does not satisfy it. Accordingly, while the synthesized system is correct, there is no guarantee about its quality. In recent years, researchers have considered extensions of the classical Boolean setting to a quantitative one. The logic LTL[$\mathcal{F}$] is a multi-valued logic that augments LTL with quality operators. The satisfaction value of an LTL[$\mathcal{F}$] formula is a real value in $[0, 1]$, where the higher the value is, the higher is the quality in which the computation satisfies the specification.

Decision problems for LTL become search or optimization problems for LTL[$\mathcal{F}$]. In particular, in the synthesis problem, the goal is to generate a transducer that satisfies the specification in the highest possible quality. Previous work considered the worst-case setting, where the goal is to maximize the quality of the computation with the minimal quality. We introduce and solve the stochastic setting, where the goal is to generate a transducer that maximizes the expected quality of a computation, subject to a given distribution of the input signals. Thus, rather than being hostile, the environment is assumed to be probabilistic, which corresponds to many realistic settings. We show that the problem is 2EXPTIME-complete, like classical LTL synthesis. The complexity stays 2EXPTIME also in two extensions we consider: one that maximizes the expected quality while guaranteeing that the minimal quality is, with probability 1, above a given threshold, and one that allows assumptions on the environment.

1998 ACM Subject Classification F.4.3 Formal Languages, B.8.2 Performance Analysis and Design Aids, F.1.1 Models of Computation

Keywords and phrases Stochastic and Quantitative Synthesis, Markov Decision Process

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.28

* A full version of the paper is available at `http://arxiv.org/abs/1608.06567`.

† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 278410, from the Israel Science Foundation (grant no. 1229/10), and from the US-Israel Binational Science Foundation (grant no. 2010431).

1 Introduction

Synthesis is the automated construction of a system from its specification: given a linear temporal logic (LTL) formula ψ over sets I and O of input and output signals, we synthesize a finite-state system that *realizes* ψ [11, 19]. At each moment in time, the system reads a truth assignment, generated by the environment, to the signals in I, and it generates a truth assignment to the signals in O. Thus, with every sequence of inputs, the system associates a sequence of outputs. The system realizes ψ if all the computations that are generated by the interaction satisfy ψ.

One weakness of automated synthesis in practice is that it pays no attention to the quality of the synthesized system. Indeed, the classical setting is Boolean: a computation satisfies a specification or does not satisfy it. Accordingly, while the synthesized system is correct, there is no guarantee about its quality. This is a crucial drawback, as designers would be willing to give up manual design only if automated-synthesis algorithms return systems of comparable quality. In recent years, researchers have considered several extensions and variants of the classical setting of synthesis. One class of extensions stays in the Boolean setting. For example, in practice we can often make assumptions on the behavior of the environment. An assumption may be direct, say given by an LTL formula that restricts the set of possible sequences of inputs [7], or conceptual, say rationality from the side of the environment, which may have its own objectives [15], or a bound on the size of the environment and/or the generated system [21, 16]. Another class of extensions moves to a quantitative setting, where a specification may have different satisfaction values in different systems. For example, in [3], the input to the synthesis problem includes also Mealy machines that grade different realizing systems. As another example, in [20], the specification formalism is the multi-valued logic LTL[$\mathcal{F}$], which augments LTL with quality operators. The satisfaction value of an LTL[$\mathcal{F}$] formula is a real value in $[0, 1]$, where the higher the value is, the higher is the quality in which the computation satisfies the specification. The synthesis algorithm then seeks systems of the highest possible quality. A quantitative approach can be taken also with Boolean specifications and involves a probabilistic view: the environment is assumed to generate input sequences according to some probability distribution. Then, instead of requiring the system to satisfy the specification in all computations generated by the environment, we measure the probability with which this happens [17].

Combining the multi-valued approach with the probabilistic one has led to the use of Markov Decision Processes (MDPs). Indeed, MDPs are a clean mathematical model that allows the analysis of quantitative objectives in a probabilistic environment. The intricacy of MDPs has led, in turn, to a plethora of works on synthesis with various constraints and reward models (e.g. [2, 6, 8, 10, 12, 1]). The starting point of these works is the MDPs. This is puzzling, as while MDPs offer a very clean framework for the analysis, they do not serve as a specification formalism. Thus, the crucial step of actually obtaining the MDPs is missing.

In this work, we consider *stochastic high-quality synthesis*, which combines the multi-valued approach with the probabilistic one. We build on known techniques for MDPs, and still keep the specification formalism accessible to designers. The specification is given by an LTL[$\mathcal{F}$] formula, the environment is assumed to be probabilistic, and we seek a system that maximizes the expected satisfaction value. To explain the setting better, let us first review shortly LTL[$\mathcal{F}$]. The linear temporal logic LTL[$\mathcal{F}$] 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 LTL[$\mathcal{F}$] is really a family of logics, each parameterized by a set $\mathcal{F} \subseteq \{f : [0, 1]^k \to [0, 1] \mid k \in \mathbb{N}\}$ of functions (of arbitrary arity)

over $[0,1]$. For example, as in earlier work on multi-valued extensions of LTL (c.f., [13]), the set $\mathcal{F}$ 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. The novelty of LTL[$\mathcal{F}$] is the ability to manipulate values by arbitrary functions. For example, $\mathcal{F}$ may contain the quantitative operator $\triangledown_\lambda$, for $\lambda \in [0,1]$, which tunes down the quality of a sub-specification. Formally, the satisfaction value of the specification $\triangledown_\lambda \varphi$ is the multiplication of the satisfaction value of φ by λ. Another useful operator is the weighted-average function $\oplus_\lambda$. There, the satisfaction value of the formula $\varphi \oplus_\lambda \psi$ is the weighted (according to λ) average between the satisfaction values of φ and ψ. This enables the quality of the system to be an interpolation of different aspects of it. As an example, consider the LTL[$\mathcal{F}$] formula $\varphi = \mathsf{G}(req \rightarrow (grant \oplus_{\frac{2}{3}} \mathsf{X} grant))$. 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 $\frac{2}{3} + \frac{1}{3} = 1$. We are quite okay with grants that are given immediately and last for only one transaction, in which case the satisfaction value is $\frac{2}{3}$, and less content when grants arrive with a delay, in which case the satisfaction value is $\frac{1}{3}$.

Consider a system that receives requests and generates grants and consider a specification ψ that has φ as defined above as a sub-formula. Other sub-formulas of ψ may require the system to generate as few grants as possible, say with $\varphi' = (\mathsf{FG}(\neg req)) \rightarrow (\mathsf{G}\neg(grant \wedge \mathsf{X} grant))$. That is, if requests eventually stop arriving, then there cannot be two successive grants. The specification ψ cannot be realized with satisfaction value 1, as the system does not know in advance whether requests eventually stops arriving. Therefore, in order to get a satisfaction value above 0 in the subformula φ', the system must not generate two successive grants, bounding the satisfaction value of the subformula φ by $\frac{2}{3}$. If, however, the input signals are distributed so that *req* may hold with a positive probability at each moment in time, then the probability that an input sequence satisfies $\mathsf{FG}(\neg req)$ is 0, causing φ' to be satisfied (that is, to have satisfaction value 1) with probability 1. Accordingly, under this assumption, a system that grants requests immediately and for two transactions has expected satisfaction value 1.

Formally, one can measure the quality of a system $\mathcal{S}$ with respect to an LTL[$\mathcal{F}$] specification taking three approaches. In the *worst-case approach*, the environment is assumed to be hostile and we care for the minimal satisfaction value of some computation of $\mathcal{S}$. In the *almost-sure approach*, the environment is assumed to be stochastic and we care for the maximal satisfaction value that is generated with probability 1. Then, in the *stochastic approach*, the environment is assumed to be stochastic and we care for the expected satisfaction value of the computations of $\mathcal{S}$, assuming some given distribution on the inputs sequences.

▶ **Example 1.** Consider a battery-replacement controller for a certain hardware. A computation of the hardware lasts k steps. Some steps during the execution are *stations*, in which the battery can be replaced. For example, the hardware may be an electric car whose battery can only be replaced at charging stations. The controller should decide at which stations it replaces the battery. On the one hand, it is wasteful to replace the battery early. On the other hand, the occurrence of stations is random, and the controller does not know whether stations are going to be encountered in the future.

Since it is wasteful to replace the battery early, the specification states that replacing it in step $1 \leq t \leq k$ lowers the satisfaction value to t/k. Missing, however, all stations incurs satisfaction value 0. We assume that each step is a station with probability $p \in [0,1]$.

Formally, the specification for the controller is over the sets $I = \{station\}$ and $O = \{replace\}$, and is a conjunction $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ of three LTL[$\mathcal{F}$] formulas (the abbreviation X^i stands for a sequence of i nested X (next) operators):

- $\varphi_1 = \mathsf{G}(\mathit{replace} \rightarrow \mathit{station})$, which requires that we only replace the battery in stations,
- $\varphi_2 = (\bigvee_{1 \leq t \leq k} \mathsf{X}^k \mathit{station}) \rightarrow (\bigvee_{1 \leq t \leq k} \mathsf{X}^k \mathit{replace})$, which states that the requirement to replace the battery needs to be satisfied only if at least one station has been encountered.
- $\varphi_3 = \bigwedge_{1 \leq t \leq k} \mathsf{X}^t(\neg \mathit{replace} \vee \nabla_{\frac{t}{k}} \mathit{replace})$, which lowers the satisfaction value to $\frac{t_0}{k}$, for the minimal step $1 \leq t_0 \leq k$ in which the battery is replaced.

In order to ensure a positive satisfaction value in the worst case, a transducer must replace the battery on the first station it encounters. Such a transducer guarantees a satisfaction value of $\frac{1}{k}$, but has expected satisfaction value of $(1-p)^k(1-\frac{1}{k}) + \frac{1}{k}$, which tends to 0 as k increases.

Trading-off the satisfaction value in the worst case for a higher expected satisfaction value, a controller may also replace the battery in later stations. For example, a transducer that replaces the battery only in the k-th step (if it is a station) has expected satisfaction value $(1-p)^k + p$. However, its satisfaction value in the worst case, in fact in $(1-p)$ of the computations, is 0.

In the full version we analyze the expected satisfaction value of a transducer that replaces the battery in the first station after position t, for $1 \leq t \leq k$, and show, for example, that a transducer that replaces the battery starting in position $\frac{k}{2}$ has an expected satisfaction value that tends to $\frac{1}{2}$ as $k \rightarrow \infty$, for every fixed $p \in (0,1)$. ◀

The worst case approach has been studied in [20], where it is shown how to synthesize, given φ, a system with a maximal worst-case satisfaction value. In this paper, we consider the two other approaches. We model a reactive system with sets I and O of input and output signals, respectively, by an I/O-transducer: a finite-state machine whose transitions are labeled by truth assignments to the signals in I and whose states are labeled by truth assignments to the signals in O. We define and solve the *stochastic high-quality synthesis* problem (SHQSyn, for short). The input to the problem is an LTL[$\mathcal{F}$] formula φ over $I \cup O$, and we seek an I/O-transducer that maximizes the expected satisfaction value of a computation, under a given distribution of the inputs. We show that the maximal expected satisfaction value is always attained by a finite-state transducer, and that computing such a transducer takes time that is doubly-exponential in φ, thus the problem is not more complex than the synthesis problem for LTL.

We continue to study two extensions of the SHQSyn problem. In the first extension, we add a lower bound on the satisfaction value that should be attained almost surely. Formally, the input to the *SHQSyn with threshold* problem is an LTL[$\mathcal{F}$] formula φ and a threshold $t \in [0,1]$, and we seek a transducer that maximizes the expected satisfaction value of φ, but such that the satisfaction value of φ in all its computations is at least t with probability 1. As we show, adding this restriction may lower the expected value. Also, our solution to the SHQSyn with threshold problem generalizes high-quality synthesis in the almost-sure approach, which we solve too. This approach has been studied for MDPs in [10, 12]. We show that while we can readily apply the existing solutions, the fact that our original specification is an LTL[$\mathcal{F}$] formula allows us to obtain slightly better solutions, with simpler analysis.

The second extension is the quantitative analogue of synthesis with environment assumptions. As discussed above, adding assumptions on the environment is a useful extension in the Boolean setting [7, 18]. In the *SHQSyn with environment assumption* problem we get as input an LTL[$\mathcal{F}$] formula φ and an environment assumption ψ, given by means of an LTL formula, and we seek a transducer that maximizes the expected satisfaction value of φ in computations that satisfy ψ. We note that the ability to reason about the quality of satisfaction in the presence of environment assumptions suggests a quantitative solution to

challenges that appear already in the Boolean setting. For example, in [4], the authors study the annoying phenomenon of systems realizing a specification by causing the assumption to fail. They suggest a synthesis algorithm that increases the cooperation between the system and its environment. Using LTL[$\mathcal{F}$], we can associate such a cooperation with high quality. We show that both extensions, of threshold and assumptions, as well as their combination, do not increase the complexity of the synthesis problem.

From a technical perspective, solving the Boolean synthesis problem amounts to translating an LTL formula to a deterministic parity automaton (DPW), viewing this automaton as a two-player parity game in which the system plays against the environment, and finding a winning strategy for the system. When the environment is assumed to be stochastic, the two-player game becomes a Markov decision process (MDP) with a parity objective. Such MDPs were extensively studied in [6, 8]. In order to handle the quantitative satisfaction values of LTL[$\mathcal{F}$], we translate an LTL[$\mathcal{F}$] formula φ to a set of DPWs associated with the different possible satisfaction values of φ. From the latter we obtain a mean-payoff MDP. We show that a transducer that attains the maximal expected satisfaction value is embodied in this MDP, and can be found in polynomial time. The analysis of the MDP is based on a search for *controllably win recurrent* states [8]. Adding a threshold $t \in [0, 1]$, the strategies of the MDP are restricted to those that guarantee that the computation reaches, with probability 1, end components that correspond to accepting runs of DPWs associated with satisfaction values above t.

Finally, in order to handle environment assumptions, we need to maximize the conditional expected satisfaction value, given the assumption. Maximizing conditional expectation is notoriously difficult, as, unlike unconditional expectation, it is not a linear objective. Thus, it is not susceptible to linear optimization techniques, which are the standard approach to find maximizing strategies in MDPs. In our solution, we compose the MDP with the DPW for the assumption, which enables us to adopt techniques used in the context of conditional probabilities in MDPs [2]. Intuitively, we add to the MDP transitions that "redistribute" the probability of computations that do not satisfy the assumption. In both cases, the size of the analyzed MDP stays doubly exponential in φ (and the assumption, in the latter case), and the required transducer is embodied in it.

Due to lack of space, some proofs are omitted and can be found in the full version, in the authors' home pages.

2 Preliminaries

2.1 Automata and Transducers

A (deterministic) *pre-automaton* is a tuple $\langle \Sigma, Q, q_0, \delta \rangle$, where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, and $\delta : Q \times \Sigma \nrightarrow Q$ is a (partial) transition function. A run of the pre-automaton on a word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is a sequence of states $q_0, q_1, q_2, \ldots$ such that $q_{j+1} = \delta(q_j, \sigma_{j+1})$ for all $j \geq 0$. Note that since δ is deterministic, the pre-automaton has at most one run on each word.

A *deterministic parity automaton* (DPW, for short) is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where $\langle \Sigma, Q, q_0, \delta \rangle$ is a pre-automaton, δ is a total function, and $\alpha : Q \to \{1, ..., d\}$ is an acceptance condition that maps states to ranks. The maximal rank d is the *index* of $\mathcal{A}$. For a run $r = q_0, q_1, q_2, \ldots$ of $\mathcal{A}$, let $\inf(r)$ be the set of states that occur in r infinitely often. Formally, $\inf(r) = \{q : q_j = q \text{ for infinitely many } j \geq 0\}$. The run r is accepting if the maximal rank of a state in $\inf(r)$ is even. Formally, $\max_{q \in \inf(r)}\{\alpha(q)\}$ is even. A word $w \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if the run of $\mathcal{A}$ on w is accepting. The language of $\mathcal{A}$, denoted

$L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts.

For finite sets I and O of input and output signals, respectively, an *I/O transducer* is $\mathcal{T} = \langle I, O, Q, q_0, \delta, \mu\rangle$, where $\langle 2^I, Q, q_0, \delta\rangle$ is a pre-automaton, and $\mu : Q \to 2^O$ is a labeling function on the states. Intuitively, $\mathcal{T}$ models the interaction of an environment that generates at each moment in time a letter in 2^I with a system that responds with letters in 2^O. Consider an input word $w = i_0 \cdot i_1 \cdots \in (2^I)^\omega$ and let $q_0, q_1, \ldots$ be the run of $\mathcal{T}$ on w. The *output* of $\mathcal{T}$ on w is then $o_1, o_2, \ldots \in (2^O)^\omega$, where $o_j = \mu(q_j)$ for all $j \geq 1$. Note that the first output assignment is that of q_1, thus $\mu(q_0)$ is ignored. This reflects the fact that the environment initiates the interaction. The *computation of $\mathcal{T}$ on w* is then $\mathcal{T}(w) = i_0 \cup o_1, i_1 \cup o_2, \ldots \in (2^{I\cup O})^\omega$.

2.2 Markov Chains and Markov Decision Processes

A *Markov chain* (MC, for short) $\mathcal{M} = \langle S, s_0, P\rangle$ consists of a finite or countably-infinite state space S, an initial state $s_0 \in S$, and a stochastic transition function $P : S \times S \to [0, 1]$. That is, for all $s \in S$, we have $\sum_{s'\in S} P(s, s') = 1$. Intuitively, when a run of $\mathcal{M}$ is in state s, then it moves to state s' with probability $P(s, s')$. A *run* of $\mathcal{M}$ is a finite or infinite sequence $s_0, s_1, s_2, ...$ of states that starts in s_0. The MC $\mathcal{M}$ induces a probability space on finite runs. Consider a finite run $r = s_0, s_1, ..., s_k$. We define $\Pr(r) = \prod_{i=1}^{k-1} P(s_i, s_{i+1})$. Thus, the probability of a finite run is the product of the probabilities of its transitions. Let $\mathrm{Cone}(r)$ be the set of all infinite runs that start with r. The MC $\mathcal{M}$ induces a probability space over the set of infinite runs of $\mathcal{M}$ that are generated by the cylinder sets $\mathrm{Cone}(r)$, for finite runs r. Formally, for every $r \in S^*$, we have $\Pr(\mathrm{Cone}(r)) = \Pr(r)$.

An *ergodic component* of $\mathcal{M}$ is a strongly connected component of $\mathcal{M}$ from which no other component is reachable. Formally, it is a set $C \subseteq S$ such that for every $s, t \in C$ there exist a path $s_1, s_2, ..., s_k$ of states in C such that $s_1 = s$, $s_k = t$, and $P(s_j, s_{j+1}) > 0$ for every $1 \leq j \leq k$. In addition, for every $s \in C$ and $t \notin C$, it holds that $P(s, t) = 0$. Let $\mathcal{C}$ be the set of maximal (w.r.t. containment) ergodic components of $\mathcal{M}$. We associate with $\mathcal{M}$ an *ergodic reachability probability* $\rho : \mathcal{C} \to [0, 1]$ such that $\rho(C)$ is the probability that a run of $\mathcal{M}$ reaches (and therefore remains forever in) C.

A *Markov decision process* (MDP) is $\mathcal{M} = \langle S, s_0, (A_s)_{s\in S}, \mathrm{P}, \gamma\rangle$, where S is a finite set of states, $s_0 \in S$ is an initial state, and A_s is a finite set of actions that are available in state $s \in S$. Let $A = \bigcup_{s\in S} A_s$. Then, $\mathrm{P} : S \times A \times S \nrightarrow [0, 1]$ is a (partial) stochastic transition function: for every two states $s, s' \in S$ and action $a \in A_s$, we have that $\mathrm{P}(s, a, s')$ is the probability of moving from s to s' when action a is taken. Accordingly, for every $s \in S$ and $a \in A_s$, we have $\sum_{s'\in S} \mathrm{P}(s, a, s') = 1$. Finally, $\gamma : S \to \mathbb{R}$ is a reward function on the states.

An MDP can be thought of as a game between a player, who chooses the action to be taken in each state, and nature, which stochastically chooses the next state according to the transition probabilities. The goal of the player is to maximize the average reward along the generated run in the MDP. We now formalize this intuition.

A *strategy* for the player in an MDP $\mathcal{M}$ (a strategy for $\mathcal{M}$, in short) is a function $f : S^+ \to A$ that suggests to the player an action to be taken given the history of the game so far. The strategy should suggest an available action, thus $f(s_0, \ldots, s_n) \in A_{s_n}$. A strategy is *memoryless* if it depends only on the current state. We can describe a memoryless strategy by $f : S \to A$, where again, $f(s) \in A_s$.

Given a strategy f, we can obtain from $\mathcal{M}$ the MC $\mathcal{M}_f = \langle S^+, s_0, P_f\rangle$ in which the choice of actions is resolved according to f. Formally, if $u, u' \in S^+$ are such that there are $t \in S^*$ and $s, s' \in S$ such that $u = t \cdot s$ and $u' = t \cdot s \cdot s'$, then $P_f(u, u') = \mathrm{P}(s, f(t \cdot s), s')$. Otherwise, $P_f(u, u') = 0$. Note that $\mathcal{M}_f$ has an infinite state space. If f is memoryless, we

can simplify the construction, and define $\mathcal{M}_f = \langle S, s_0, P_f \rangle$ with $P_f(s, s') = \mathrm{P}(s, f(s), s')$.

An *end component* in an MDP $\mathcal{M}$ is a set $C \subseteq S$ such that there exist action sets $(B_s)_{s \in S}$ with $B_s \subseteq A_s$ for every $s \in S$, and for every $s, t \in C$, there exists a path $s_1, s_2, ..., s_k$ of states in C such that $s_1 = s$, $s_k = t$ and there exist actions $a_1, ..., a_{k-1}$ such that $\mathrm{P}(s_j, a_i, s_{j+1}) > 0$ and $a_i \in B_{s_j}$ for every $1 \leq j \leq k$. In addition, for every $s \in C$ and $a \in B_s$ it holds that $\sum_{t \in C} \mathrm{P}(s, a, t) = 1$. Intuitively, an end component is a strongly-connected component in the MDP graph that nature cannot force to leave. Equivalently, $\mathcal{M}$ has a strategy to stay forever in C. Indeed, it is not hard to see that C is an end component iff there is some strategy f for $\mathcal{M}$ such that C is an ergodic component of $\mathcal{M}_f$.

The *value* $\mathrm{val}_{\mathcal{M}}(f)$ (we omit the subscript when $\mathcal{M}$ is clear from context) of a strategy f for $\mathcal{M}$ is the expected average reward of an infinite run in $\mathcal{M}_f$. Formally, for a run $r = s_0, s_1, s_2, \ldots$ of $\mathcal{M}_f$, we define $\gamma(r) = \liminf_{m \to \infty} \frac{1}{m} \sum_{j=0}^{m} \gamma(s_j)$, where for a state $s \in S^+$ of $\mathcal{M}_f$, the cost $\gamma(s)$ is induced by the last state of $\mathcal{M}$ in s. In the stochastic setting, we view each sequence of inputs, and hence also each run r and the reward on r, as a random variable. The *expected value* of a random variable is, intuitively, its average value, weighted by probabilities. Let $R_{\mathcal{M},f}$ be the random variable whose value is the reward on runs in $\mathcal{M}_f$. We define $\mathrm{val}_{\mathcal{M}}(f) = \mathbb{E}[R_{\mathcal{M},f}]$. The *value* $\mathrm{val}(\mathcal{M})$ of an MDP $\mathcal{M}$ is the maximal value of a strategy in $\mathcal{M}$. It is well known (see e.g. [14]) that $\mathrm{val}(\mathcal{M})$ can be attained by a memoryless strategy, which can be computed in polynomial time.

For technical reasons, we sometimes use variants of MDPs. A *pre-MDP* is an MDP with no reward function. A *parity MDP* is a pre-MDP with a parity acceptance condition $\alpha : S \to \{1, ..., d\}$. In a parity MDP, the goal of the player is to maximize the probability that the generated run satisfies the parity condition. Parity-MDPs were extensively studied in e.g. [9].

2.3 The logic LTL[$\mathcal{F}$]

The logic LTL[$\mathcal{F}$] is a multi-valued logic that extends the linear temporal logic LTL with an arbitrary set of functions $\mathcal{F} \subseteq \{f : [0,1]^k \to [0,1] : k \in \mathbb{N}\}$ called quality operators. For example, $\mathcal{F}$ may contain the maximum or minimum between the satisfaction values of subformulas, their product, and their average. This enables the specifier to refine the Boolean correctness notion and associate different possible ways of satisfaction with different truth values [20].

Let AP be a set of Boolean atomic propositions and let $\mathcal{F}$ be a set of function as described above. 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 $f \in \mathcal{F}$.

The semantics of LTL[$\mathcal{F}$] formulas is defined with respect to infinite computations over $2^{I \cup O}$. For a computation $\pi = \pi_0, \pi_1, \ldots \in (2^{I \cup O})^\omega$ and position $j \geq 0$, we use π^j to denote the suffix $\pi_j, \pi_{j+1}, \ldots$. 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 in $[0,1]$ and is defined inductively as described in Table 1 below.

The logic LTL can be viewed as LTL[$\mathcal{F}$] for $\mathcal{F}$ that models the usual Boolean operators. For simplicity, we use the common such functions as abbreviation, as described below. In addition, we introduce notations for two useful quality operators, namely factoring and weighted average. Let $x, y, \lambda \in [0,1]$. Then,

- $\neg x = 1 - x$
- $x \vee y = \max\{x, y\}$
- $x \wedge y = \min\{x, y\}$

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

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	$[\![\pi, \mathsf{X}\varphi_1]\!]$	$[\![\pi^1, \varphi_1]\!]$
$[\![\pi, p]\!]$	$\begin{matrix}1 & \text{if } p \in \pi_0\\ 0 & \text{if } p \notin \pi_0\end{matrix}$	$[\![\pi, \varphi_1 \mathsf{U} \varphi_2]\!]$	$\max_{0 \le i < \lvert\pi\rvert} \{\min\{[\![\pi^i, \varphi_2]\!], \min_{0 \le j < i} [\![\pi^j, \varphi_1]\!]\}\}$

- $x \to y = \max\{1 - x, y\}$
- $\nabla_\lambda x = \lambda \cdot x$
- $x \oplus_\lambda y = \lambda \cdot x + (1 - \lambda) \cdot y$

▶ **Example 2.** Consider a scheduler that receives requests and generates grants and consider the LTL[$\mathcal{F}$] formula $\varphi = \varphi_1 \wedge \varphi_2$, with $\varphi_1 = \mathsf{G}(req \to \mathsf{X}(grant \oplus_{\frac{2}{3}} \mathsf{X}grant))$ and $\varphi_2 = \neg(\nabla_{\frac{3}{4}} \mathsf{G}\neg req)$. The satisfaction value of the formula φ_1 is 1 if every request is granted in the next cycle and the grant lasts for two consecutive cycles. If the grant lasts for only one cycle, then the satisfaction value is reduced to $\frac{2}{3}$ if it is the cycle right after the request, and to $\frac{1}{3}$ if it is the next one. In addition, the conjunction with φ_2 implies that if there are no requests, then the satisfaction value is at most $\frac{1}{4}$. The example demonstrates how LTL[$\mathcal{F}$] can conveniently prioritize different scenarios, as well as embody vacuity considerations in the formula. ◀

For an LTL[$\mathcal{F}$] formula φ, let $V(\varphi) = \{[\![\pi, \varphi]\!] : \pi \in (2^{AP})^\omega\}$. That is, $V(\varphi)$ is the set of possible satisfaction values of φ in arbitrary computations.

▶ **Theorem 3** ([20]). *Consider an* LTL[$\mathcal{F}$] *formula* φ.

- $|V(\varphi)| \le 2^{|\varphi|}$.
- *For every predicate* $\theta \subseteq [0, 1]$, *there exists a DPW* $\mathcal{A}_{\varphi,\theta}$ *such that* $L(\mathcal{A}_{\varphi,\theta}) = \{\pi : [\![\pi, \varphi]\!] \in \theta\}$. *Furthermore,* $\mathcal{A}_{\varphi,\theta}$ *has at most* $2^{2^{O(|\varphi|)}}$ *states and its index is at most* $2^{|\varphi|}$.

3 High-Quality Synthesis

Consider an I/O-transducer $\mathcal{T}$ and an LTL[$\mathcal{F}$] formula φ over $I \cup O$. Each computation of $\mathcal{T}$ may have a different satisfaction value for φ. We can measure the quality of $\mathcal{T}$ taking three approaches:

- *Worst-case approach:* The environment is assumed to be hostile and we care for the minimal satisfaction value of some computation of $\mathcal{T}$. Formally, $[\![\mathcal{T}, \varphi]\!]_w = \min\{[\![\mathcal{T}(w), \varphi]\!] : w \in (2^I)^\omega\}$. Note that no matter what the input sequence is, the specification φ is satisfied with value at least $[\![\mathcal{T}, \varphi]\!]_w$.
- *Almost-sure approach:* The environment is assumed to be stochastic and we care for the maximal satisfaction value that is generated with probability 1. Formally, given a distribution ν of $(2^I)^\omega$, we define $[\![\mathcal{T}, \varphi]\!]^\nu_a = \max\{v :$ there is W with $\nu(W) = 1$ and $[\![\mathcal{T}(w), \varphi]\!] \ge v$ for every $w \in W\}$. Note that the specification φ is satisfied almost surely with value at least $[\![\mathcal{T}, \varphi]\!]^\nu_a$.
- *Stochastic approach:* The environment is assumed to be stochastic and we care for the expected satisfaction value of the computations of $\mathcal{T}$, assuming some given distribution on the inputs sequences. Formally, let $X_{\mathcal{T},\varphi} : (2^I)^\omega \to \mathbb{R}$ be a random variable that assigns to each sequence $w \in (2^I)^\omega$ of input signals the value $[\![\mathcal{T}(w), \varphi]\!]$. Then, given a distribution ν of $(2^I)^\omega$, we define $[\![\mathcal{T}, \varphi]\!]^\nu_s = \mathbb{E}[X_{\mathcal{T},\varphi}]$, when the sequences in $(2^I)^\omega$ are sampled according to ν.

The worst case approach has been studied in [20], where it is shown how to find $[\![\mathcal{T},\varphi]\!]_w$ and how to synthesize, given φ, a transducer with a maximal worst-case satisfaction value. In this paper, we consider the stochastic approach. For simplicity, we consider environments with a uniform distribution on the input signals. That is, ν is such that at each moment in time, each input signal holds with probability $\frac{1}{2}$, thus the probability of each letter in 2^I is $\frac{1}{2^{|I|}}$ (see Remark 4). Since ν is fixed, we omit it from the notation and use $[\![\mathcal{T},\varphi]\!]_a$ and $[\![\mathcal{T},\varphi]\!]_s$.

▶ Remark 4 (On the choice of a uniform distribution). Recall that we consider a uniform distribution on the letters in 2^I. In practice, the distribution on the truth assignments to the input signals may be richer. In the general case, such a distribution can be given by an MDP.

Adjusting our setting and algorithms to handle such distributions involves only a small technical elaboration, orthogonal to the technical challenges that exist already in the setting of a uniform distribution. Accordingly, throughout the paper we assume a uniform distribution. In Section 7.2, we describe how our setting and algorithms are extended to the general case. ◀

▶ **Example 5.** Consider a hard-drive writing protocol that needs to finalize a write operation through some connection. The connection needs to be closed as soon as possible, to allow access to the drive. However, data may still arrive in the first two cycles, and if the connection is closed in the first cycle, then the data that arrives in the second cycle gets lost. The issue is that the decision as to whether to close the connection is made during the first cycle, before the protocol knows whether data is going to arrive in the second cycle. The specification that formulates the above scenario is over $I = \{data\}$ and $O = \{close\}$ and is $\varphi = ((\mathsf{X}data) \rightarrow \neg close) \wedge ((\neg \mathsf{X}data) \rightarrow close) \vee \nabla_{\frac{1}{2}} \mathsf{X}close)$.

That is, if data arrives in the second cycle, then we should not close the connection in the first cycle. In addition, if data does not arrive in the second cycle, we should close the connection in the first cycle – this would give us satisfaction value 1 in the second conjunct, but we may also close the connection only in the second cycle, which would guarantee a satisfaction value of 1 in the first conjunct, but would reduce the satisfaction value of the second conjunct to $\frac{1}{2}$ in cases data does not arrive in the second cycle.

Let $p \in [0,1]$ be the probability that data arrives in the second cycle. Consider a transducer $\mathcal{T}_1$ that closes the connection in the first cycle. With probability p, we have that $\mathsf{X}data$ holds, in which case φ has satisfaction value 0. Also, with probability $1-p$, we have that $\mathsf{X}data$ does not hold and the satisfaction value of φ is 1. Thus, the satisfaction value of φ is 0 in the worst case, and this is also the highest satisfaction value that $\mathcal{T}_1$ achieves with probability 1. On the other hand, the expected satisfaction value of φ in a computation of $\mathcal{T}_1$ is $p \cdot 0 + (1-p) \cdot 1 = 1-p$. Thus, $[\![\mathcal{T}_1,\varphi]\!]_w = [\![\mathcal{T}_1,\varphi]\!]_a = 0$, whereas $[\![\mathcal{T}_1,\varphi]\!]_s = 1-p$.

Consider now a transducer $\mathcal{T}_2$ that closes the connection only on the second cycle. With probability p, we have that $\mathsf{X}data$ holds, in which case the satisfaction value of φ is 1. Also, with probability $1-p$, we have that $\mathsf{X}data$ does not hold, in which case the satisfaction value of φ is $\frac{1}{2}$. Thus, now the satisfaction value of φ is $\frac{1}{2}$ in the worst case, and this is also the highest satisfaction value that $\mathcal{T}_2$ achieves with probability 1. On the other hand, the expected satisfaction value of φ in a computation of $\mathcal{T}_2$ is $p \cdot 1 + (1-p) \cdot \frac{1}{2} = \frac{1}{2}(1+p)$. Thus, $[\![\mathcal{T}_2,\varphi]\!]_a = [\![\mathcal{T}_2,\varphi]\!]_w = \frac{1}{2}$, whereas $[\![\mathcal{T}_2,\varphi]\!]_s = \frac{1}{2}(1+p)$.

To conclude, when $p \geq \frac{1}{3}$, in which case $\frac{1}{2}(1+p) \geq 1-p$, then $\mathcal{T}_2$ is superior to $\mathcal{T}_1$ in all the three approaches. When, however, $p < \frac{1}{3}$, then a designer that cares for the expected satisfaction value should prefer $\mathcal{T}_1$. ◀

3.1 The Achievability MDP of an LTL[$\mathcal{F}$] formula

In this section we develop the technical tool we are going to use for solving the high-quality synthesis problem in the stochastic approach.

Consider an LTL[$\mathcal{F}$] formula φ. Let $V(\varphi) = \{v_1, ..., v_n\}$, with $v_1 < ... < v_n \in [0, 1]$. By Theorem 3, we have that $n \leq 2^{|\varphi|}$. Also, for every $1 \leq i \leq n$, there is a DPW $\mathcal{A}_i$ such that $L(\mathcal{A}_i) = \{w : [\![w, \varphi]\!] = v_i\}$. Let $\mathcal{A}_i = \langle 2^{I \cup O}, Q^i, q_0^i, \delta^i, \alpha^i \rangle$. We construct the product pre-automaton $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ that subsumes the joint behavior of the DPWs. Formally, $\mathcal{A} = \langle 2^{I \cup O}, S, s_0, \mu \rangle$, where $S = Q^1 \times \ldots \times Q^n$, the initial state is $s_0 = \langle q_0^1, ..., q_0^n \rangle$, and for every state $s = \langle q_1, ..., q_n \rangle$ and $\sigma \in 2^{I \cup O}$, we have $\mu(s, \sigma) = \langle \delta^1(q_1, \sigma), ..., \delta^n(q_n, \sigma) \rangle$.

Every pre-automaton $\mathcal{B} = \langle 2^{I \cup O}, Q, q_0, \delta \rangle$ induces a pre-MDP $\mathcal{M}_\mathcal{B} = \langle Q, q_0, 2^O, \mathrm{P} \rangle$ in which for every two states $q, q' \in S$ and action $o \in 2^O$, we have $\mathrm{P}(q, o, q') = \frac{|\{i \in 2^I : \delta(q, i \cup o) = q'\}|}{2^{|I|}}$. That is, choosing action $o \in 2^O$ in state q, the MDP samples the possible inputs $i \in 2^I$ uniformly and moves to state $\delta(q, i \cup o)$. Consider a memoryless strategy $f : Q \to 2^O$ for $\mathcal{M}_\mathcal{B}$. The strategy f induces an I/O-transducer $\mathcal{T}[\mathcal{M}_\mathcal{B}, f] = \langle I, O, Q, q_0, \delta', \mu \rangle$ in which for every state $q \in Q$, we have $\mu(q) = f(q)$, and for all $i \in 2^I$, we have $\delta'(q, i) = \delta(i \cup \mu(q))$. Thus, the transducer has the same state space as $\mathcal{B}$, it lets f fix the labels of the states, and uses this label to complete the 2^I component of the alphabet to a letter in $2^{I \cup O}$.

Consider a parity acceptance condition α on the state space Q of $\mathcal{B}$. Using the notations of [9], a state $q \in Q$ in $\mathcal{M}_\mathcal{B}$ is *controllably win recurrent* (c.w.r., for short) if there exists an end component $U \subseteq Q$ such that $q \in U$, $\alpha(q) = \max_{p \in U} \{\alpha(p)\}$, and $\alpha(q)$ is even. That is, q has the maximal rank in U, and this rank is even. The end component U is referred to as a *witness* for q being c.w.r. Intuitively, a parity-MDP with a parity objective α has a strategy to win with probability 1 from all c.w.r. states. Moreover, if U is a witness for some c.w.r. state, then there exists a strategy to win with probability 1 from every state in U. If, however, a run of $\mathcal{M}_\mathcal{B}$ reaches, and stays forever, in an end component that does not have a c.w.r. state, then it is winning with probability 0.

Once we have defined the product pre-automaton $\mathcal{A}$, we construct an MDP $\mathcal{M}_\mathcal{A} = \langle S, s_0, 2^O, \mathrm{P}, \gamma \rangle$, with the following reward function. For a state $s = \langle q_1, ..., q_n \rangle$ of $\mathcal{M}_\mathcal{A}$, we say that a value $v_i \in V(\varphi)$ is *achievable* from s if there exists a c.w.r. state in $\mathcal{M}_{\mathcal{A}_i}$ with a witness U_i for which $q_i \in U_i$. Then, $\gamma(s) = \max\{v_i : v_i \text{ is achievable from } s\}$. Note that the way we have defined $\mathcal{A}$ guarantees that every state that is a part of some end component has at least one value v_i that is achievable from s. For states that are not in end components, we define the reward to be 0. Intuitively, $\gamma(s)$ is the highest satisfaction value that can be guaranteed with probability 1 from s. We refer to $\mathcal{M}_\mathcal{A}$ as the *achievability MDP for* φ.

This completes the construction of $\mathcal{M}_\mathcal{A}$. Note that every end component U consists of states with the same value v_U. Thus, every infinite run r of $\mathcal{M}$ eventually gets trapped in some end component U, implying that $\gamma(r) = v_U$. Indeed, the rewards along the states in the finite prefix of r that leads to U are averaged out. For an end-component U of $\mathcal{M}_\mathcal{A}$, let $U|_i$ be the projection of U on Q^i. Note that $U|_i$ is an end component in $\mathcal{A}_i$.

4 Synthesis Against a Stochastic Environment

In the *stochastic high-quality synthesis* problem (SHQSyn, for short), we get as input an LTL[$\mathcal{F}$] formula φ over sets I and O of input and output signals, and we seek an I/O-transducer that maximizes the expected value of a computation (under a uniform distribution of the inputs). Formally, we want to compute $\max_\mathcal{T} \{\mathbb{E}[[\![\mathcal{T}, \varphi]\!]_s]\}$ and return the witness

transducer.[1]

We solve the SHQSyn problem by reasoning on the achievable MDP $\mathcal{M}_{\mathcal{A}}$. Consider a strategy f for $\mathcal{M}_{\mathcal{A}}$. Let $\mathcal{T}$ be the transducer induced from f, that is $\mathcal{T} = \mathcal{T}[M_{\mathcal{A}}, f]$. Recall the random variable $X_{\mathcal{T},\varphi} : (2^I)^\omega \to \mathbb{R}$ that maps $w \in (2^I)^\omega$ to $[\![\mathcal{T}(w), \varphi]\!]$. We define the random variables $Y_{\mathcal{T},\varphi} : (2^I)^\omega \to \mathbb{R}$ as follows. For every $w \in (2^I)^\omega$, we let $Y_{\mathcal{T},\varphi}(w)$ be the mean-payoff of the values along the run of $\mathcal{A}$ on $\mathcal{T}(w)$. Formally, let r be the run of $\mathcal{A}$ on $\mathcal{T}(w)$. Then, $Y_{\mathcal{T},\varphi}(w) = \gamma(r)$, where γ is the reward function of $\mathcal{M}_{\mathcal{A}}$. By definition, we have that $[\![\mathcal{T}, \varphi]\!]_s = \mathbb{E}[X_{\mathcal{T},\varphi}]$. Since $\mathcal{M}_{\mathcal{A}}$ is obtained by assuming a uniform distribution on the inputs, we have that $\mathbb{E}[Y_{\mathcal{T},\varphi}] = \mathrm{val}_{\mathcal{M}_{\mathcal{A}}}(f)$.

▶ **Theorem 6.** *Consider an* LTL[$\mathcal{F}$] *formula* φ*. Let* $\mathcal{M}_{\mathcal{A}}$ *be the achievability MDP for* φ*. For every value* $v \in [0,1]$*, there exists a strategy* f *in* $\mathcal{M}_{\mathcal{A}}$ *such that* $val_{\mathcal{M}_{\mathcal{A}}}(f) \geq v$ *iff there exists an* I/O*-transducer* $\mathcal{T}$ *such that* $[\![\mathcal{T}, \varphi]\!]_s \geq v$*. Moreover, we can find in time polynomial in* $\mathcal{M}_{\mathcal{A}}$ *a memoryless strategy* f *such that* $[\![\mathcal{T}[\mathcal{M}_{\mathcal{A}}, f], \varphi]\!]_s$ *maximizes* $\{\mathbb{E}[[\![\mathcal{T}, \varphi]\!]_s]\}$*.*

Proof. We start by proving that if there exists a transducer $\mathcal{T}$ such that $[\![\mathcal{T}, \varphi]\!]_s \geq v$, then there exists a strategy f such that $\mathrm{val}_{\mathcal{M}_{\mathcal{A}}}(f) \geq v$. For this, we prove, in the full version, that $\mathbb{E}[X_{\mathcal{T},\varphi}] \leq \mathbb{E}[Y_{\mathcal{T},\varphi}]$. This is indeed sufficient, as we can then take f to be the strategy induced by $\mathcal{T}$.

For the converse implication, consider a strategy f in $\mathcal{M}_{\mathcal{A}}$ such that $\mathrm{val}_{\mathcal{M}_{\mathcal{A}}}(f) \geq v$. By [14], we can assume that f is memoryless. Let $\mathcal{T} = \mathcal{T}[\mathcal{M}_{\mathcal{A}}, f]$ be the transducer induced by f. In the full version, we show that there exists a transducer $\mathcal{T}'$ such that $\mathbb{E}[X_{\mathcal{T}',\varphi}] = \mathbb{E}[Y_{\mathcal{T},\varphi}]$, thus concluding the claim. ◀

We now proceed to show how to solve the SHQSyn problem.

▶ **Theorem 7.** *Solving the SHQSyn problem for* LTL[$\mathcal{F}$] *can be done in doubly-exponential time. The corresponding decision problem is 2EXPTIME complete.*

Proof. Consider an LTL[$\mathcal{F}$] formula φ. We want to find a transducer $\mathcal{T}$ that maximizes $[\![\mathcal{T}, \varphi]\!]_s$. Let $\mathcal{M}_{\mathcal{A}}$ be the achievability MDP for φ. By Theorem 6, we can find in time polynomial in $\mathcal{M}_{\mathcal{A}}$ a memoryless strategy f such that $[\![\mathcal{T}[\mathcal{M}_{\mathcal{A}}, f], \varphi]\!]_s$ maximizes $\{\mathbb{E}[[\![\mathcal{T}, \varphi]\!]_s]\}$. Below we analyze the size of $\mathcal{M}_{\mathcal{A}}$. Let $|\varphi| = k$. By Theorem 3, we have that $n \leq 2^k$ and each $\mathcal{A}_i$ is of size at most $2^{2^{O(k)}}$. Thus, the size of $\mathcal{M}_{\mathcal{A}}$ is at most $(2^{2^{O(k)}})^{2^k} = 2^{2^{O(k)}}$, implying the doubly exponential upper bound.

A matching lower bound for the respective decision problem follows from the 2EXPTIME hardness of standard LTL synthesis. Note that in our setting one considers satisfaction with probability 1. Still, since the hardness proof for LTL synthesis considers the interaction between a system and its environment along a finite prefix of a computation (one that models the computation of a Turing machine that halts), it applies also for the stochastic setting. ◀

5 Adding an Almost-Sure Threshold

In this section we combine the stochastic and the almost-sure approaches. The *SHQSyn problem with a threshold* includes both an LTL[$\mathcal{F}$] formula φ and a threshold $t \in [0,1]$. The goal then is to maximize the expected satisfaction value of φ while guaranteeing that it is almost surely above t. Formally, given φ and t, we seek a transducer $\mathcal{T}$ that maximizes

$$\{[\![\mathcal{T}, \varphi]\!]_s : [\![\mathcal{T}, \varphi]\!]_a \geq t\}.$$

[1] A-priori, it is not clear that the maximum is attained. As we prove, however, this is in fact the case.

Note that there need not be a transducer $\mathcal{T}$ for which $[\![\mathcal{T}, \varphi]\!]_a \geq t$, in which case the set is empty and we return no transducer. This is the multi-valued analogue of an unrealizable Boolean specification (except that here the user may want to try to reduce t). Note also that this sub-problem, of deciding whether the set is empty, amounts to solving the high-quality synthesis problem in the almost-sure approach. Finally, if the set is not empty, then we have to show, as in Section 4, that its maximum is indeed attained.

▶ **Example 8.** Consider a server sending messages over a noisy channel. At each cycle, the server sends a message and needs to decide whether to encode it so that error-correction can retrieve it in case the channel is noisy, or take a risk and send the message with no encoding. Encoding a message has some cost. We formulate the quality of each cycle by the specification ψ over $I = \{noise\}$ and $O = \{encode\}$, where $\psi = (\neg noise \wedge \neg encode) \vee \nabla_{\frac{3}{4}} encode$. Thus, each cycle has satisfaction value 1 if a message that is not encoded is sent over a non-noisy channel, and satisfaction value $\frac{3}{4}$ if a message is encoded. Note that otherwise (that is, when a message that is not encoded is sent over a noisy channel), the satisfaction value is 0. The factor $\frac{3}{4}$ in the LTL[$\mathcal{F}$] specification reflects the priorities of the designer as induced by the actual cost of encoding and of losing messages.

Recall that ψ specifies the quality of a single cycle. The quality of a full computation refers to its different cycles, and a natural thing to do is to take the average over the cycles we want to consider. Assume that a channel may be noisy only during the first four cycles. Then, the quality of a computation is $\varphi = (\psi \oplus_{\frac{1}{2}} \mathsf{X}\psi) \oplus_{\frac{1}{2}} (\mathsf{XX}\psi \oplus_{\frac{1}{2}} \mathsf{XXX}\psi)$.

Assume that the probability of a channel to be noisy in each of the first four cycles is p. Consider a transducer $\mathcal{T}_1$ that does not encode any message. The expected satisfaction value of ψ in each of the four cycles is then $(1-p) \cdot 1 + p \cdot 0 = 1 - p$, hence $[\![\mathcal{T}_1, \varphi]\!]_s = 1 - p$. On the other hand, the satisfaction value of ψ in a noisy cycle is 0, hence $[\![\mathcal{T}_1, \varphi]\!]_w = [\![\mathcal{T}_1, \varphi]\!]_a = 0$. Thus, if one does not care for a lower bound on the satisfaction value in the worst case, then by using $\mathcal{T}_1$ he gets an expected satisfaction value of $1 - p$.

Suppose now that we want the satisfaction value to be above $\frac{1}{3}$ in the worst case. This can be achieved by a transducer $\mathcal{T}_2$ that encodes messages in two of the four cycles. Indeed, for the cycles in which a message is encoded, we get satisfaction value $\frac{3}{4}$, which is averaged with 0, namely the worst-case satisfaction value in the cycles in which a message is not encoded. Hence, $[\![\mathcal{T}_2, \varphi]\!]_w = [\![\mathcal{T}_2, \varphi]\!]_a = \frac{3}{4} \oplus_{\frac{1}{2}} 0 = \frac{3}{8} > \frac{1}{3}$. The expected satisfaction value of $\mathcal{T}_2$ is then $[\![\mathcal{T}_2, \varphi]\!]_s = \frac{3}{4} \oplus_{\frac{1}{2}} (1-p) = \frac{7}{8} - \frac{p}{2}$.

Finally, if we want to ensure satisfaction value $\frac{3}{4}$ in the worst case, then we can design a transducer $\mathcal{T}_3$ that encodes all the messages in the first four cycles. Now, $[\![\mathcal{T}_3, \varphi]\!]_w = [\![\mathcal{T}_3, \varphi]\!]_a = [\![\mathcal{T}_3, \varphi]\!]_s = \frac{3}{4}$.

It follows that for a small p, adding a threshold on the satisfaction value in the worst case reduces the expected satisfaction value. Indeed, when $p < \frac{1}{4}$, then $1 - p > \frac{7}{8} - \frac{p}{2} > \frac{3}{4}$. When, however, $p \geq \frac{1}{4}$, then $\mathcal{T}_3$ is superior in the three approaches. ◀

In order to solve the SHQSyn problem with a threshold, we modify our solution from Section 3.1 as follows. We start by deciding whether there exists a transducer $\mathcal{T}$ such that $[\![\mathcal{T}, \varphi]\!]_a \geq t$. For this, we construct, per Theorem 3, a DPW $\mathcal{A}_{\geq t} = \langle 2^{I \cup O}, Q^{\geq t}, q_0^{\geq t}, \delta^{\geq t}, \alpha^{\geq t} \rangle$ such that $L(\mathcal{A}_{\geq t}) = \{w : [\![w, \varphi]\!] \geq t\}$. Let $\mathcal{M}_{\geq t}$ be the parity-MDP induced from $\mathcal{A}_{\geq t}$. By [9], we can find the set of almost-sure winning states of $\mathcal{M}_{\geq t}$. If $q_0^{\geq t}$ is winning, then the required transducer exists, and in fact $\mathcal{M}_{\geq t}$ embodies all candidate transducers. We obtain a pre-automaton $\mathcal{A}'_{\geq t}$ from $\mathcal{A}_{\geq t}$ by restricting $\mathcal{A}_{\geq t}$ to winning states, and removing transitions from state $q \in Q^{\geq t}$ for every action $o \in 2^O$ such that there exists $i \in 2^I$ for which $\delta^{\geq t}(q, i \cup o)$ is not a winning state.

We proceed by constructing a product pre-automaton $\mathcal{A}$ that is similar to the one constructed in Section 3.1, except that takes t and $\mathcal{A}'_{\geq t}$ into account, as follows.

Let $\ell = \arg\min_i \{v_i : v_i \geq t\}$ be the minimal index such that $v_i \geq t$. We define $\mathcal{A} = \mathcal{A}_\ell \times \ldots \times \mathcal{A}_n \times \mathcal{A}'_{\geq t}$. That is, the product, defined as in Section 3.1, now contains only DPWs $\mathcal{A}_i$ for which $v_i \geq t$ and also contains $\mathcal{A}'_{\geq t}$. We obtain the MDP $\mathcal{M}_\mathcal{A}$ and set the reward function as in Section 3.1, taking into account only c.w.r. states from the automata $\mathcal{A}_\ell, ..., \mathcal{A}_n$. The component $\mathcal{A}'_{\geq t}$ is only used to restrict the actions of the MDP $\mathcal{M}_\mathcal{A}$. We refer to $\mathcal{M}_\mathcal{A}$ as the *t-achievability MDP for φ*.

We present an analogue to Theorem 6. The proof appears in the full version.

▶ **Theorem 9.** *Consider an* LTL[$\mathcal{F}$] *formula φ and a threshold $t \in [0,1]$. Let $\mathcal{M}_\mathcal{A}$ be the t-achievability MDP for φ. For every value $v \in [0,1]$, there exists a strategy f in $\mathcal{M}_\mathcal{A}$ such that $val_{\mathcal{M}_\mathcal{A}}(f) \geq v$ iff there exists an I/O-transducer $\mathcal{T}$ such that $[\![\mathcal{T}, \varphi]\!]_a \geq t$ and $[\![\mathcal{T}, \varphi]\!]_s \geq v$. Moreover, we can find in time polynomial in $\mathcal{M}_\mathcal{A}$ a memoryless strategy f such that $[\![\mathcal{T}[\mathcal{M}_\mathcal{A}, f], \varphi]\!]_s$ maximizes $\{\mathbb{E}[[\![\mathcal{T}, \varphi]\!]_s] : [\![\mathcal{T}, \varphi]\!]_a \geq t\}$.*

Since, by Theorem 3, the size of $\mathcal{A}_{\geq t}$ is doubly exponential in φ, then, by following considerations similar to those specified in the proof of Theorem 7, we conclude with the following.

▶ **Theorem 10.** *Solving the SHQSyn problem with a threshold for* LTL[$\mathcal{F}$] *can be done in doubly-exponential time. The corresponding decision problem is 2EXPTIME-complete.*

▶ Remark 11. In [10, 12], the authors solve the problem of deciding, given an MDP $\mathcal{M}$ and two thresholds v and t, whether there is a strategy f for $\mathcal{M}$ that guarantees value t almost surely, and has expected cost at least v. The solution can be directly applied to our setting. However, note that this solution only guarantees an expected cost of v, whereas our approach finds the optimal expected cost. ◀

▶ Remark 12. In the SHQSyn problem with a threshold, we use the formula φ both for the expectation maximization, and for the almost-sure threshold. Sometimes, it is desirable to decompose the specification into one part – ψ, which is a hard constraints and needs to be satisfied almost-surely above the threshold t, and another part – φ, which specifies a utility function with respect to which we would like to optimize [5, 12].

Our solution can be easily adapted to handle this setting. Indeed, in the construction of the t-achievability MDP, we replace $\mathcal{A}_{\geq t}$, with $\mathcal{B}_{\geq t}$, where $L(\mathcal{B}_{\geq t}) = \{w : [\![w, \psi]\!] \geq t\}$, and proceed with the described construction and the proofs. ◀

6 Adding Environment Assumptions

A common paradigm in Boolean synthesis is synthesis with environment assumptions [7, 18], where the input to the synthesis problem consists of a specification φ and an assumption ψ, and we seek a transducer that realizes φ under the assumption that the environment satisfies ψ. In this section we consider an analogue variant of the SHQSyn problem, where we are given an LTL[$\mathcal{F}$] specification φ and an LTL assumption ψ, and we seek a transducer that maximizes the expected satisfaction value of φ given that the environment satisfies the assumption ψ. Note that while the specification is quantitative, the assumption is Boolean.

▶ **Example 13.** Recall the message-sending server in Example 8, and assume that the channel can change its status (noisy/non-noisy) only every second cycle. We use this assumption in order to design improved transducers. Formally, the assumption is given by the LTL formula $\psi = (noise \leftrightarrow \mathsf{X} noise) \wedge \mathsf{XX}(noise \leftrightarrow \mathsf{X} noise)$.

The transducer $\mathcal{T}_4$ does not encode the first message, but checks whether the channel was noisy. If it was, the second message is encoded. We get that the expected satisfaction value of φ in $\mathcal{T}_4$ under the assumption is $(1-p+p\cdot\frac{3}{4}+(1-p)\cdot 1)/2 = 1-\frac{5}{8}p$, which is higher than $1-p = [\![\mathcal{T}_1,\varphi]\!]_s$ for every $p>0$. In addition, under the assumption we are guaranteed that the worst-case satisfaction value of $\mathcal{T}_4$ is at least $\frac{3}{8}$, unlike $\mathcal{T}_1$ (in case the channel is noisy, so only the second and fourth messages are encoded). Thus $\mathcal{T}_4$ is superior to $\mathcal{T}_1$ described in Example 8 in the three approaches (under the assumption).

Next, as in Example 8, if we want to ensure satisfaction value $\frac{3}{4}$ in the worst case, we can design a transducer $\mathcal{T}_5$ that works like $\mathcal{T}_4$, except that it always encodes the first and third messages. The expected satisfaction value of $\mathcal{T}_5$ under the assumption is $(\frac{3}{4}+p\cdot\frac{3}{4}+(1-p)\cdot 1)/2 = \frac{7}{8}-\frac{p}{8}$, which is higher than $\frac{3}{4} = [\![\mathcal{T}_3,\varphi]\!]_s$, for every $p\in[0,1]$.

Thus, under the assumption, it is possible to design transducers that increase the expected satisfaction value as well as the lower bound. ◀

Formally, in the SHQSyn problem with environment assumptions, we get as input an LTL[$\mathcal{F}$] formula φ over $I\cup O$, and an *environment assumption* ψ, which is an LTL formula over I such that $\Pr(\psi)>0$. That is, the probability of the event $\{w : w\models\psi\}\subseteq(2^I)^\omega$ is strictly positive. Recall that $X_{\mathcal{T},\varphi}$ is a random variable such that $X_{\mathcal{T},\varphi}(w) = [\![\mathcal{T}(w),\varphi]\!]$. We seek a transducer $\mathcal{T}$ that maximizes $\mathbb{E}[X_{\mathcal{T},\varphi}|w\models\psi]$.

We start by citing a folklore lemma, whose proof can be found in the full version.

▶ **Lemma 14.** *Consider a random variable X. Let A,B be events such that $\Pr(A)>0$ and $\Pr(B)=0$. Then, $\mathbb{E}[X|A\cup B] = \mathbb{E}[X|A]$.*

Before proceeding, we note that if $\Pr(\psi)=1$, then we can proceed by dropping the assumption entirely. Indeed, it holds that $\Pr(\neg\psi)=0$, and by Lemma 14, we have that $\mathbb{E}[X_{\mathcal{T},\varphi}|w\models\psi] = \mathbb{E}[X_{\mathcal{T},\varphi}|(w\models\psi)\cup(w\models\neg\psi)] = \mathbb{E}[X_{\mathcal{T},\varphi}|(2^I)^\omega] = \mathbb{E}[X_{\mathcal{T},\varphi}]$. Thus, we henceforth assume that $0<\Pr(\psi)<1$.

As mentioned in Section 1, maximizing the conditional expectation directly is notoriously problematic, as, unlike unconditional expectation, it is not a linear objective. Thus, it is not susceptible to linear optimization techniques, which are the standard approach to find maximizing strategies in MDPs. Our solution is a modification of the construction from Section 3.1 in which we, intuitively, "redistribute" the probability of the input sequences that do not satisfy the assumption. We start by constructing a DPW $\mathcal{A}_\psi$ that accepts a word $w\in(2^I)^\omega$ iff $w\models\psi$. Note that the alphabet of $\mathcal{A}$ is 2^I. We think of this alphabet as $2^{I\cup O}$, where transitions simply ignore the 2^O component. In particular, the MDP $\mathcal{M}_{\mathcal{A}_\psi}$ is in fact an MC. We say that an ergodic component of $\mathcal{M}_{\mathcal{A}_\psi}$ is *rejecting* if the maximal rank that appears in it is odd. It is easy to see that a run in a rejecting ergodic component is accepting w.p. 0.

We then consider the automaton $\mathcal{A} = \mathcal{A}_\psi\times\mathcal{A}_1\times\ldots\times\mathcal{A}_n$, and obtain the MDP $\mathcal{M}_\mathcal{A} = \langle S, s_0, 2^O, \mathrm{P}, \gamma\rangle$ as described in Section 3.1. In particular, the reward function is as there, and the only change is the addition of the $\mathcal{A}_\psi$ component, which provides information about satisfaction of ψ. We refer to $\mathcal{M}_\mathcal{A}$ as the *conditional achievability MDP for φ given ψ*. Recall that for a strategy f, we have defined $R_{\mathcal{M}_A,f}$ as a random variable whose value is the reward on runs in $\mathcal{M}_\mathcal{A}$ with strategy f. Following the proof of Theorem 6, we then get the following.

▶ **Theorem 15.** *Consider an* LTL[$\mathcal{F}$] *formula φ and an environment assumption ψ. Let $\mathcal{M}_\mathcal{A}$ be the conditional achievability MDP for φ given ψ. For every value $v\in[0,1]$, there exists a strategy f in $\mathcal{M}_\mathcal{A}$ such that $\mathbb{E}[R_{\mathcal{M}_A,f}|w\models\psi]\geq v$ iff there exists an I/O-transducer*

$\mathcal{T}$ such that $\mathbb{E}[X_{\mathcal{T},\varphi}|w \models \psi] \geq v$. Moreover, if f is memoryless, then we can find in time polynomial in $\mathcal{M}_\mathcal{A}$ a memoryless strategy f such that $\mathbb{E}[X_{\mathcal{T}[\mathcal{M}_\mathcal{A},f],\varphi}|w \models \psi] \geq v$.

Theorem 15 enables us to reason about $\mathcal{M}_\mathcal{A}$, but we are still left with conditional expectations. To handle the latter, we follow a technique suggested in [2] and obtain from $\mathcal{M}_\mathcal{A}$ a new MDP $\mathcal{M}'_\mathcal{A} = \langle S, s_0, A, \mathrm{P}', \gamma\rangle$ as follows. A state $s = \langle q, q_1, ..., q_n\rangle$ of $\mathcal{M}_\mathcal{A}$ is called a *rejecting ergodic state* if its state q of $\mathcal{A}_\psi$ belongs to a rejecting ergodic component of $\mathcal{M}_{\mathcal{A}_\psi}$. Let $\mathscr{R} = \{s : s \text{ is a rejecting ergodic state}\}$.

For every state $s \in \mathscr{R}$ we set $\mathrm{P}'(s, a, s_0) = 1$. That is, whenever a rejecting ergodic component of $\mathcal{A}_\psi$ is reached, the MDP $\mathcal{M}'_\mathcal{A}$ deterministically resets back to s_0.

Intuitively, when a rejecting ergodic component of $\mathcal{A}_\psi$ is reached, then the probability of ψ being satisfied is 0. Thus, resetting "redistributes" the probability of ψ not being satisfied evenly. Below we formalize this intuition. The proofs can be found in the full version.

▶ **Lemma 16.** *Let $v \in \mathbb{R}$, and consider a memoryless strategy g in $\mathcal{M}'_\mathcal{A}$ such that $val_{\mathcal{M}'_\mathcal{A}}(g) \geq v$. There exists a memoryless strategy f in $\mathcal{M}_\mathcal{A}$ such that $\mathbb{E}[R_{\mathcal{M}_\mathcal{A},f}|w \models \psi] \geq v$. Moreover, f can be computed from g in polynomial time.*

▶ **Lemma 17.** *Let $v \in \mathbb{R}$, and consider a strategy f in $\mathcal{M}_\mathcal{A}$ such that $\mathbb{E}[R_{\mathcal{M}_\mathcal{A},f}|w \models \psi] \geq v$. There exists a strategy g in $\mathcal{M}'_\mathcal{A}$ such that $\mathbb{E}[R_{\mathcal{M}'_\mathcal{A},g}] \geq v$.*

Finally, using Theorem 15, and the fact that $\mathcal{A}_\psi$ is doubly exponential in ψ, we can use the same reasoning as in the proof of Theorem 7 and conclude with the following. The proof can be found in the full version.

▶ **Theorem 18.** *Solving the SHQSyn problem with environment assumptions can be done in doubly-exponential time. The corresponding decision problem is 2EXPTIME-complete.*

7 Extensions

In this section we describe two extensions to the setting. The first combines the threshold and assumption extensions presented in Sections 5 and 6. The second shows how to handle a non-uniform probability distribution.

7.1 Combining an Almost-Sure Threshold with Environment Assumptions

Combining an almost-sure threshold with environment assumptions requires some subtlety in the definitions. As an input for the problem, we are given an LTL[$\mathcal{F}$] formula φ over $I \cup O$, an LTL environment assumption ψ over I such that $\Pr(\psi) > 0$, and a threshold $t \in [0, 1]$. Then, we seek a a transducer $\mathcal{T}$ that maximizes $\mathbb{E}[X_{\mathcal{T},\varphi}|w \models \psi]$ and for which $\Pr([\![\mathcal{T}(w), \varphi]\!] \geq t|w \models \psi) = 1$. In particular, the threshold t should be attained almost surely only in computations that satisfy ψ.

▶ Remark 19. Note that it could have also been possible to seek a transducer $\mathcal{T}$ that maximizes $\mathbb{E}[X_{\mathcal{T},\varphi}|w \models \psi]$ and for which $\Pr([\![\mathcal{T}(w), \varphi]\!] \geq t) = 1$, namely for which the threshold should hold almost surely regardless of the assumption. We found this approach less appealing. Its solution, however, is a straightforward combination of our constructions. That is, we start with the product $\mathcal{A}_\ell \times \ldots \times \mathcal{A}_n \times \mathcal{A}'_{\geq t} \times \mathcal{A}_\psi$, as defined in Sections 5 and 6, apply the reset modification described in Section 6, and seek a maximizing strategy in the resulting MDP. ◀

We solve the problem as follows. We start by checking whether there exists a transducer $\mathcal{T}$ such that $\Pr([\![\mathcal{T}(w), \varphi]\!] \geq v | w \models \psi) = 1$, using the following lemma (see the full version for the proof).

▶ **Lemma 20.** *Let φ, ψ, and t be as above. For every transducer $\mathcal{T}$ it holds that* $\Pr([\![\mathcal{T}(w), \varphi]\!] \geq t | w \models \psi) = 1$ *iff* $\Pr([\![\mathcal{T}(w), \psi \rightarrow \varphi]\!] \geq t) = 1$.

Using Lemma 20, we can decide the existence of a transducer $\mathcal{T}$ as we seek, by constructing the DPW $\mathcal{A}_{\psi \rightarrow \varphi, \geq t}$ as per Theorem 3, and keeping only almost-sure winning states as done in Section 5.

We now proceed as in the first approach, by constructing the product $\mathcal{A}_\ell \times \ldots \times \mathcal{A}_n \times \mathcal{A}'_{\psi \rightarrow \varphi, \geq t} \times \mathcal{A}_\psi$, where $\mathcal{A}'_{\psi \rightarrow \varphi, \geq t}$ is obtained from $\mathcal{A}_{\psi \rightarrow \varphi, \geq t}$ by keeping only almost-sure winning states.

7.2 Handling a Non-Uniform Distribution

In order to handle a non-uniform distribution on the input signals, we first have to decide how to model arbitrary distributions on $(2^I)^\omega$. The common way to do so is to assume that the distribution is generated by a pre-MDP $\mathcal{D} = \langle S, s_0, 2^O, \mathrm{P} \rangle$ and a labeling function $\iota : S \rightarrow 2^I$, where a state $s \in S$ generates the input $\iota(s)$. Thus, the probability of an input signal to hold depends on the history of the interaction with the system. Formally, every run $r = s_0, s_1, ...$ of $\mathcal{D}$ generates an input sequence $\iota(s_1), \iota(s_2)$, and the distribution on runs induces a distribution on $(2^I)^\omega$. [2]

All our results can be adapted to handle a distribution given by $\mathcal{D}$ as above. We only have to change the construction of the achievability MDP described in Section 3.1 as follows. For a pre-automaton $\mathcal{B} = \langle 2^{I \cup O}, Q, q_0, \delta \rangle$ and a distribution pre-MDP $\mathcal{D} = \langle S, s_0, 2^O, \mathrm{P} \rangle$ with labeling function ι, we define the induced pre-MDP as $\mathcal{M}^{\mathcal{D}}_{\mathcal{B}} = \langle Q \times S, \langle q_0, s_0 \rangle, 2^O, \mathrm{P}' \rangle$ where for every two states $\langle q, s \rangle, \langle q', s' \rangle \in Q \times S$ and action $o \in 2^O$, we have $\mathrm{P}'(\langle q, s \rangle, o, \langle q', s' \rangle) = \mathrm{P}(s, o, s')$ if $\delta(q, \iota(s') \cup o) = q'$, and 0 otherwise. It is not hard to see that all the constructions we apply to achievability MDP $\mathcal{M}_{\mathcal{A}}$ can be applied to $\mathcal{M}^{\mathcal{D}}_{\mathcal{A}}$, which would take the distribution in $\mathcal{D}$ into account. The complexity of the algorithms is polynomial in $\mathcal{M}^{\mathcal{D}}_{\mathcal{A}}$. Thus, the complexity of our algorithms remains 2EXPTIME-complete in φ and polynomial in $\mathcal{D}$.

References

1. S. Almagor, O. Kupferman, and Y. Velner. Minimizing expected cost under hard boolean constraints, with applications to quantitative synthesis. In *27th CONCUR*, 2016.
2. C. Baier, J. Klein, S. Klüppelholz, and S. Märcker. Computing conditional probabilities in markovian models efficiently. In *20th TACAS*, pages 515–530, 2014.
3. R. Bloem, K. Chatterjee, T. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *21st CAV*, volume 5643 of *LNCS*, pages 140–156, 2009.
4. R. Bloem, R. Ehlers, and R. Könighofer. Cooperative reactive synthesis. In *13th ATVA*, pages 394–410, 2015.
5. V. Bruyère, E. Filiot, M. Randour, and J-F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. In *31st STACS*, volume 25 of *LIPIcs*, pages 199–213, 2014.
6. K. Chatterjee and L. Doyen. Energy and mean-payoff parity markov decision processes. In *36th MFCS*, pages 206–218, 2011.

[2] Note that we do not consider the label on s_0, in order to allow a distribution on the initial letters.

7 K. Chatterjee, T. Henzinger, and B. Jobstmann. Environment assumptions for synthesis. In *19th CONCUR*, volume 5201 of *LNCS*, pages 147–161, 2008.

8 K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Games with secure equilibria. In *19th LICS*, pages 160–169, 2004.

9 K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Quantitative stochastic parity games. In *SODA 04*, pages 121–130, 2004.

10 K. Chatterjee, Z. Komárková, and J. Kretínský. Unifying two views on multiple mean-payoff objectives in markov decision processes. In *30th LICS*, pages 244–256, 2015.

11 A. Church. Logic, arithmetics, and automata. In *Proc. Int. Congress of Mathematicians, 1962*, pages 23–35. Institut Mittag-Leffler, 1963.

12 L. Clemente and J-F. Raskin. Multidimensional beyond worst-case and almost-sure problems for mean-payoff objectives. In *30th LICS*, pages 257–268, 2015.

13 M. Faella, A. Legay, and M. Stoelinga. Model checking quantitative linear time logic. *Electr. Notes Theor. Comput. Sci.*, 220(3):61–77, 2008.

14 J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1996.

15 D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *16th TACAS*, volume 6015 of *LNCS*, pages 190–204, 2010.

16 O. Kupferman, Y. Lustig, M.Y. Vardi, and M. Yannakakis. Temporal synthesis for bounded systems and environments. In *28th STACS*, pages 615–626, 2011.

17 M. Kwiatkowska and D. Parker. Automated verification and strategy synthesis for probabilistic systems. In *11th ATVA*, volume 8172 of *LNCS*, pages 5–22, 2013.

18 W. Li, L. Dworkin, and S. A. Seshia. Mining assumptions for synthesis. In *9th MEMOCODE*, pages 43–50, 2011.

19 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *16th POPL*, pages 179–190, 1989.

20 U. Boker S. Almagor and and O. Kupferman. Formalizing and reasoning about quality. *J. ACM*, 63(3), 2016.

21 S. Schewe and B. Finkbeiner. Bounded synthesis. In *5th ATVA*, volume 4762 of *LNCS*, pages 474–488, 2007.