Minimizing Expected Cost Under Hard Boolean Constraints, with Applications to Quantitative Synthesis

Shaull Almagor, Orna Kupferman, and Yaron Velner

School of Computer Science and Engineering, The Hebrew University, Israel.

Abstract. In Boolean synthesis, we are given an LTL specification, and the goal is to construct a transducer that realizes it against an adversarial environment. Often, a specification contains both Boolean requirements that should be satisfied against an adversarial environment, and multi-valued components that refer to the quality of the satisfaction and whose expected cost we would like to minimize with respect to a probabilistic environment.

In this work we study, for the first time, mean-payoff games in which the system aims at minimizing the expected cost against a probabilistic environment, while surely satisfying an ω-regular condition against an adversarial environment. We consider the case the ω-regular condition is given as a parity objective or by an LTL formula. We show that in general, optimal strategies need not exist, and moreover, the limit value cannot be approximated by finite-memory strategies. We thus focus on computing the limit-value, and give tight complexity bounds for synthesizing ϵ-optimal strategies for both finite-memory and infinite-memory strategies.

We show that our game naturally arises in various contexts of synthesis with Boolean and multi-valued objectives. Beyond direct applications, in synthesis with costs and rewards to certain behaviors, it allows us to compute the minimal sensing cost of ω-regular specifications – a measure of quality in which we look for a transducer that minimizes the expected number of signals that are read from the input.

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 system that *realizes* ψ [11,18]. 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 extensions of the classical Boolean setting to a quantitative one, which takes quality into account.

Quality measures can refer to the system itself, examining parameters like its size or its consumption of memory, sensors, voltage, bandwidth, etc., or refer to the way the system satisfies the specification. In the latter, we allow the designer to specify the quality of a behavior using quantitative specification formalisms [1,5,13]. For example, rather than the Boolean specification requiring all requests to be followed by a grant, a quantitative specification formalism would give a different satisfaction value to a computation in which requests are responded immediately and one in which requests are responded after long delays.[1]

Solving the synthesis problem in the Boolean setting amounts to solving a two-player zero-sum game between the system and the environment. The goal of the system is to satisfy the (Boolean) specification, and the environment is adversarial. Then, a winning strategy for the system corresponds to a transducer that realizes the specification. In the quantitative setting, the goal of the system is no longer Boolean, as every play is assigned a cost by the specification. In the classical quantitative approach, we measure the satisfaction value in the worst-case semantics. Thus, the value of a strategy for the system is the maximal cost of a play induced by this strategy, and the goal of the system is to minimize this value. Recently, there is a growing interest also in the expected cost of a play, under a probabilistic environment. The motivation behind this approach is that the quality of satisfaction is a “soft constraint”, and should not be measured in a worst-case semantics. Then, the game above is replaced by a mean-payoff Markov Decision Process (MDP): a game in which each state has a cost, inducing also costs to infinite plays (essentially, the cost of an infinite play is the limit of the average cost of prefixes of increased lengths). The goal is to find a strategy that minimizes the expected cost [10,12].

While quantitative satisfaction refines the Boolean one, often a specification contains both Boolean conditions that should be satisfied against all environments, and multi-valued components that refer to the quality of the satisfaction and whose expectation we would like to minimize with respect to a probabilistic environment. Accordingly, researchers have suggested the *beyond worst-case* approach, where a specification has both hard and soft constraints, and the goal is to realize the hard constraints, while maximizing the expected satisfaction value of the soft constraints. In Section 1.1 below, we describe this approach and related work in detail.

In this work, we consider, for the first time, mean-payoff MDPs equipped with a parity winning condition (parity-MDPs, for short). The goal is to find a strategy that surely wins the parity game (that is, against an adversarial environment), while minimizing the expected cost of a play against a probabilistic environment. While the starting point in earlier related work is the MDP itself, possibly augmented by different objectives, our starting point depends on the application, and we view the construction of the MDP as an integral part of our contribution. We focus on two applications: synthesis with penalties to undesired scenarios and synthesis with minimal sensing.

[1] Note that the polarity of some quality measures is negative, as we want to minimize size, consumption, costs, etc., whereas the polarity of other measures is positive, as we want to maximize performance and satisfaction value. For simplicity, we assume that all measures are associated with costs, which we want to minimize.

Let us describe the two applications. We start with penalties to scenarios. Consider an LTL specification ψ over I and O. Activating an output signal may have a cost; for example, when the activation involves a use of a resource. Taking these costs into account, the input to the synthesis problem includes, in addition to ψ, a cost function γ assigning cost to some assignments to output signals. The cost of a computation is then the mean cost of assignments in it. While the specification ψ is a hard constraint, as we only allow correct computations, minimizing the expected cost of computations with respect to γ is a soft constraint. Assignments correspond to scenarios of length one. More elaborated cost functions refer to on-going regular scenarios. Power consumption, for example, is an important consideration in modern chip design, from portable servers to large server farms. As the chips become more complex, the cost of powering a server farm can easily outweigh the cost of the servers themselves, thus design teams go to great lengths in order to reduce power consumption in their designs. The most widely researched logical power saving techniques are *clock gating*, in which a clock is prevented from making a "tick" if it is redundant (c.f., [4]), and *power gating*, in which whole sections of the chip are powered off when not needed and then powered on again [15,14]. The goal of these techniques is to reduce power consumption and the number of changes in the values of signals, the main source of power consumption in chips. The input to the problem of synthesis with penalties to scenarios includes, in addition to ψ, a set of deterministic automata on finite words, each describing a undesired scenario and its cost. For example, it is easy to specify the scenario of "value flip" with a two-state deterministic automaton. We show how the setting can be easily translated into solving our parity-MDPs, thus generating systems that realize ψ with minimal expected cost.

Our primary application considers activation of sensors. The quality measure of sensing was introduced in [2,3], as a measure for the detail with which a random input word needs to be read in order to realize the specification. In the context of synthesis, our goal is to construct a transducer that realizes the specification and minimizes the expected average number of sensors (of input signals) that are used along the interaction. Thus, the hard constraint in the LTL specification, and the soft one is the expected number of active sensors. Giving up sensing has a flavor of synthesis with incomplete information [16]: the transducer has to realize the specification no matter what the incomplete information is. Thus, as opposed to the examples above, the modeling of cost involves a careful construction of the MDP to be analyzed, and also involves an exponential blow-up, which we show to be unavoidable. In [3], the problem was solved for safety specifications. Our solution to the parity-MDP problem enables a solution for full LTL. We also study the complexity of the problem when the input is an LTL formula, rather than a deterministic automaton.

Back to parity-MDPs, we show that in general, optimal strategies need not exist. That is, there are parity-MDPs in which an infinite-state strategy can get arbitrarily close to some limit optimal value, but cannot attain it. Moreover, the limit value cannot be approximated by finite-memory strategies. Accordingly, our solution to parity-MDPs suggests two algorithms. The first, described in Section 3.1, finds the limit value of all possible strategies, which corresponds to infinite-state transducers. The second, described in Section 3.2, computes the limit value over all finite-memory strategies. The complexity of both algorithms is NP∩coNP. Moreover, they are computable in polynomial time when an oracle to a two-player parity game is given. Hence, our complexity upper bounds match the trivial lower bounds that arise from the fact that every solution to a parity-MDP is also a solution to a parity game. For our applications, we show that the complexity of the synthesis problem for LTL specifications stays doubly-exponential, as in the Boolean setting, even when we minimize penalties to undesired scenarios or minimize sensing.

1.1 Related Work

The combination of worst-case synthesis with expected-cost synthesis, dubbed *beyond worst-case synthesis*, was studied in [6,12] for models that are closely related to ours. In [6] the authors study mean-payoff MDPs, where both the hard constraints and the soft constraints are quantitative. Thus, a system needs to ensure a strict upper bound on the mean-payoff cost, while minimizing the expected cost. In [12], multidimensional mean-payoff MDPs are considered. Thus, the MDP is equipped with several mean-payoff costs, and the goal is to find a system that ensures the mean-payoff in some of the mean-payoffs is below an upper bound, while minimizing the expected mean-payoffs (or rather, approximating their Pareto-curve).

In comparison, our work is the first to consider a hard Boolean constraint (namely the parity condition). This poses both a conceptual and a technical difference. Conceptually, when quantitative synthesis is taken as a refinement of Boolean synthesis, it is typically meant as a ranking of different systems that satisfy a Boolean specification. Thus, it makes sense for the hard constraint to be Boolean as well. Technically, combining Boolean and quantitative constraints gives rise to some subtleties that do not exist in the pure-quantitative setting. Specifically, when both the hard and the soft constraints are quantitative, a strategy can intuitively “alternate” between satisfying them. Thus, if while trying to meet the soft constraint the hard constraint is violated, we can switch to a worst-case strategy until the hard constraint is satisfied, and go back to trying to minimize the soft constraint. This alternation can be done infinitely often. In the Boolean setting, however, this alternation can violate the Boolean constraint. We note that unlike classical parity games, where the parity winning condition can be translated to a richer mean-payoff objective, the parity winning condition in our parity-MDPs does not admit a similar translation.

Other works on MDPs and mean-payoff objectives tackle different aspects of quantitative analysis. In [19], a solution to the expected mean-payoff value over MDPs is presented. In [7] and [8], the authors study a combination of mean-payoff and parity objectives over MDPs and over stochastic two-player games. There, the goal of the system is to ensure with probability 1 that the parity condition holds and that the mean-payoff is below a threshold. This differs from our work in that the parity condition is not a hard constraint, as it is met only almost-surely, and in that the expected mean-payoff is not guaranteed to be minimized. As detailed in the paper, these differences make the technical challenges very different.

Due to lack of space, some proofs appear in the appendix.

2 Parity-MDPs

A *parity Markov decision process* (Parity-MDP, for short) combines a parity game with a mean-payoff MDP. The game is played between Player 1, who models a system, and Player 2, who models the environment. The environment is adversarial with respect to the parity winning condition and is stochastic with respect to the mean-payoff objective. Formally, a parity-MDP is a tuple $\mathcal{M} = \langle S_1, S_2, s_0, A_1, A_2, \delta_1, \delta_2, \mathrm{P}, cost, \alpha \rangle$, with the following components. The sets S_1 and S_2 are finite set of states, for Players 1 and 2, respectively. Let $S = S_1 \cup S_2$. Then, $s_0 \in S$ is an initial state, and A_1 and A_2 are sets of actions for the players. Not all actions are available in all states: for every state $s \in S_i$, for $i \in \{1, 2\}$, we use $A_i(s)$ to denote the finite set of actions available to Player i in the state s. For $i \in \{1, 2\}$, the transition function $\delta_i : S_i \times A_i \nrightarrow S$ is such that $\delta_i(s, a)$ is defined iff $a \in A_i(s)$. Let $\delta = \delta_1 \cup \delta_2$. Note that δ_2 gets an action of Player 2 as a parameter. We distinguish between two approaches to the way the action is chosen. In the adversarial approach, it is Player 2 who chooses the action. In the stochastic approach, the choice depends on the (partial) function $\mathrm{P} : S_2 \times A_2 \nrightarrow [0, 1]$, where for every state $s \in S_2$ and $a \in A_2$, we have that $\mathrm{P}(s, a) > 0$ only if $a \in A_2(s)$. Also, $\sum_{a \in A_2(s)} \mathrm{P}(s, a) = 1$. Finally, $cost : S \rightarrow \mathbb{N}$ is a cost function, and $\alpha : S \rightarrow \{0, ..., d\}$, for some $d \in \mathbb{N}$, is a parity winning condition.

The parity-MDP $\mathcal{M}$ induces a *parity game* $\mathcal{M}^{\mathtt{P}} = \langle S_1, S_2, s_0, A_1, A_2, \delta_1, \delta_2, \alpha \rangle$, obtained by omitting P and $cost$. In this game, we follow the adversarial approach to the environment. Thus, both players choose their actions. Formally, a *strategy* for Player i in $\mathcal{M}$, for $i \in \{1, 2\}$ is a function $f_i : S^* \times S_i \rightarrow A_i$ such that for $s_0, \ldots, s_n$, we have $f(s_0, \ldots, s_n) \in A_i(s_n)$. Thus, a strategy suggests to Player i an available action given the history of the states traversed so far. Note that we do not consider randomized strategies, but rather deterministic ones. Our results in Section 3 show that this is sufficient, in the sense that the players cannot gain by using randomization.

Given strategies f_1 and f_2 for Players 1 and 2, the *play* induced f_1 and f_2 is is the infinite sequence of states $s_0, s_1, ...$ such that for every $j \geq 0$, if $s_j \in S_i$, for $i \in \{1, 2\}$, then $s_{j+1} = \delta_i(s_j, f(s_0, ..., s_j))$. For an infinite play r, we denote by inf(r) the set of states that r visits infinitely often. The play $r = s_0, s_1, ...$ of $\mathcal{M}$ is *parity winning* if $\max \{\alpha(s) : s \in \text{inf}(r)\}$ is even.

The parity-MDP $\mathcal{M}$ also induces an *MDP* $\mathcal{M}^{\mathtt{MDP}} = \langle S_1, S_2, s_0, A_1, A_2, \delta_1, \delta_2, \mathrm{P}, cost \rangle$, obtained by omitting α. In this game, we follow the stochastic approach to the environment and consider the distribution of plays when only a strategy for Player 1 is given. Formally, we first extend P to transitions as follows: For states $s \in S_2$ and $s' \in S$, we define $\mathrm{P}(s, s') = \sum_{a \in A(s) : \delta_2(s,a) = s'} \mathrm{P}(s, a)$. Then, a play of $\mathcal{M}$ with strategy f_1 for Player 1 is an infinite sequence of states $s_0, s_1, ...$ such that for every $j \geq 0$, if $s_j \in S_1$, then $s_{j+1} = \delta_1(s_j, f_1(s_0, ..., s_j))$, and if $s_j \in S_2$, then $\mathrm{P}(s_j, s_{j+1}) > 0$. The *cost* of a strategy f_1 is the expected average cost of a random walk in $\mathcal{M}$ in which Player 1 proceeds according to f_1. Formally, for $m \in \mathbb{N}$ and for a prefix $\tau = s_0, s_1, ...s_m$ of a play, let $I_2 = \{j : j < m \text{ and } s_j \in S_2\}$. Then, we define $\mathrm{P}_f(\tau) = \prod_{j \in I} \mathrm{P}(s_j, s_{j+1})$ and $cost_m(f, \tau) = \frac{1}{m+1} \sum_{j=0}^{m} cost(s_j)$. The cost of a strategy f_1 is then $cost(f_1) = \liminf_{m \to \infty} \sum_{\tau : |\tau| = m} cost_m(f_1, \tau) \cdot \mathrm{P}_f(\tau)$. We denote by inf$(f)$ the random variable that associates inf(ρ) with a sequence of states $\rho = s_0, s_1, ...$, under the probability space induced by $\mathcal{M}$ with f.

A *finite memory* strategy for $\mathcal{M}$ is described by a finite set M called *memory*, an initial memory $init \in M$, a *memory update function* $next : S_1 \times M \to M$, and an *action function* $act : S_1 \times M \to A_1$ such that $act(s, m) \in A_1(s)$ for every $s \in S_1$ and $m \in M$.

A strategy is *memoryless* if it has finite memory M with $|M| = 1$. Note that a memoryless strategy depends only on the current state. Thus, we can describe a memoryless strategy by $f_1 : S_1 \to A_1$. Let $cost(\mathcal{M}) = \inf\{cost(f_1) : f_1$ is a strategy for $\mathcal{M}\}$. That is, $cost(\mathcal{M})$ is the expected cost of a game played on $\mathcal{M}$ in which Player 1 uses an optimal strategy.

The following is a basic property of MDPs.

Theorem 1. *Consider an MDP $\mathcal{M}$. Then, $cost(\mathcal{M})$ can be attained by a memoryless strategy, which can be computed in polynomial time.*

Recall that a strategy f_1 for player 1 is winning in M^{P} if every play of $\mathcal{M}$ with f_1 satisfies the parity condition α. Note that we require *sure* winning, in the sense that all plays must be winning, rather than winning with probability 1 (*almost-sure* winning). On the other hand, the definition of cost in M^{MDP} considered strategies for Player 1 and ignore the parity winning condition. We now define the *sure cost* of the parity-MDP, which does take them into account. For a strategy f_1 for Player 1, the sure cost of f_1, denoted $cost_{\text{sure}}(f_1)$, is $cost(f_1)$, if f_1 is winning, and is ∞ otherwise. The sure cost of $\mathcal{M}$ is then $cost_{\text{sure}}(\mathcal{M}) = \inf\{cost_{\text{sure}}(f_1) : f_1$ is a strategy for $\mathcal{M}\}$.

End Components Consider a parity-MDP $\mathcal{M} = \langle S_1, S_2, s_0, A_1, A_2, \delta_1, \delta_2, \mathrm{P}, cost, \alpha\rangle$. An *end component* (EC, for short) is a set $U \subseteq S$ such that for every state $s \in U$, the following hold.

1. If $s \in S_1$, then there exists an action $a \in A_s$ such that $\delta_1(s, a) \in U$.
2. If $s \in S_2$, then for every $a \in A_2(s)$ such that $\mathrm{P}(s, a) > 0$, it holds that $\delta_2(s, a) \in U$.
3. For every $t, t' \in U$, there exist a path $t = t_0, t_1, ..., t_k = t'$ and actions $a_1, ..., a_t$ such that for every $0 \leq i < t$, it holds that $t_i \in U$, and there exists an action a such that $\delta(t_i, a) = t_{i+1}$.

Intuitively, the probabilistic player cannot force to leave U, and Player 1 has positive probability of reaching every state in U from every other state.

For an EC U and a state $s \in U$, we can consider the parity-MDP $\mathcal{M}|_U^s$, in which the states are U, the initial state is s, and all the components are naturally restricted to U. Since U is an EC, then this is indeed a parity-MDP. An EC U is *maximal* if for every nonempty $U' \subseteq S \setminus U$, we have that $U \cup U'$ is not an EC.

3 Solving Parity MDPs

In this section we study the problem of finding the sure cost for an MDP. Recall that for MDPs, there always exists an optimal memoryless strategy. We start by demonstrating that for the sure cost of parity-MDPs, the situation is much more complicated.

Theorem 2. *There is a parity-MDP $\mathcal{M}$ in which Player 1 does not have an optimal strategy (in particular, not a memoryless one) for attaining the sure cost of $\mathcal{M}$. Moreover, for every $\epsilon > 0$, Player 1 may need infinite memory in order to ϵ-approximate $cost_{\text{sure}}(\mathcal{M})$.*

Proof. Consider the parity-MDP $\mathcal{M}$ appearing in Figure 1. Player 1 can decrease the cost of a play towards 1 by staying in the initial state. However, in order to ensure an even parity rank, Player 1 must either play b and reach a states with parity rank 2 and cost 10 w.p. 0.5, or play c but incur cost 10. A finite memory strategy for Player 1 must eventually play c from the initial state in every play,[2] thus the cost of every winning finite-memory strategy is 10. On the other hand, for every $\epsilon > 0$, there exists an infinite memory strategy f that gets cost at most $1 + \epsilon$. Essentially (see Lemma 1 for a formal proof of the general case), the strategy f plays b for a long time. If the state with parity rank 2 is reached, it plays b for even longer, and otherwise plays c.

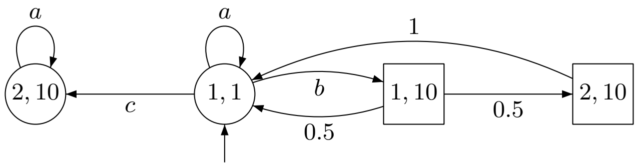

Fig. 1. The Parity MDP $\mathcal{M}$. States of Player 1 are circles, these of Player 2 are squares, with outgoing edges marked by their probability. Each state is labeled by its parity rank (left) and cost (right). Player 1 has no optimal strategy and needs infinite memory for an ϵ approximation.

Finally, there is no optimal strategy for Player 1, as every strategy that plays c from the initial state eventually (i.e., as a response to some strategy of Player 2) gets cost 10 with some positive probability. However, a strategy that never plays c is not parity-winning.

Following Theorem 2, our solution to parity MDPs suggests two algorithms. The first, described in Section 3.1, finds the limit value of all possible strategies, which corresponds to infinite-state transducers. The second, described in Section 3.2, computes the limit value over all finite-memory strategies. The complexity of both algorithms is NP∩coNP. Moreover, they are computable in polynomial time when an oracle to a two-player parity game is given. Hence, our complexity upper bounds match the trivial lower bounds that arise from the fact that every solution to a parity-MDP is also a solution to a parity game.

3.1 Infinite-Memory Strategies

In this section we study the problem of finding the sure cost of a parity-MDP when infinite-memory strategies are allowed. We prove the upper bound in the following theorem. As stated above, the lower bound is trivial.

Theorem 3. *Consider a parity-MDP $\mathcal{M}$. Then, $cost_{\mathrm{sure}}(\mathcal{M})$ can be computed in NP∩co-NP, and is parity-games hard.*

[2] Note that this also implies that randomized strategies could not be of help here.

Consider a parity-MDP $\mathcal{M} = \langle S_1, S_2, s_0, A_1, A_2, \delta_1, \delta_2, \mathrm{P}, cost, \alpha \rangle$. We first remove from $\mathcal{M}$ all states that are not sure-winning for Player 1 in $\mathcal{M}^{\mathtt{P}}$. Clearly, every strategy that attains $cost_{\mathrm{sure}}(\mathcal{M})$ cannot visit a state that is losing in $\mathcal{M}^{\mathtt{P}}$. Thus, we henceforth assume that all states in $\mathcal{M}$ are winning for Player 1 in $\mathcal{M}^{\mathtt{P}}$. We say that an EC C of $\mathcal{M}$ is *good* (GEC, for short) if its maximal rank is even. That is, $\max_{s \in C} \{\alpha(s)\}$ is even.

The idea behind our algorithm is as follows. W.p. 1, each play in $\mathcal{M}$ eventually reaches and visits infinitely often all states of some EC. Hence, when restricting attention to plays that are winning for Player 1 in $\mathcal{M}^{\mathtt{P}}$, it must be the case that this EC is good. It follows that the sure cost of $\mathcal{M}$ is affected only by the properties of its GECs. Moreover, since the minimal expected mean-payoff value is the same in all the states of an EC, we can consider only maximal GECs and refer to the value of an EC, namely the minimal expected value that Player 1 can ensure while staying in the EC. Our algorithm constructs a new MDP (without ranks) $\mathcal{M}'$ in which the cost of a state is the value of the maximal GEC it belongs to. If a state does not belong to a GEC, then we assign it a very high cost in $\mathcal{M}'$, where the intuition is that Player 1 cannot benefit from visiting this state infinitely often. We claim that the sure cost in the parity-MDP $\mathcal{M}$ coincides with the cost of the MDP $\mathcal{M}'$.

Formally, for an EC C, let $C^{\max}$ be the set of the states of C with the maximal parity rank in C. By definition, this rank is even when C is a GEC. Note that if C and C' are GECs and $C \cap C' \neq \emptyset$, then $C \cup C'$ is also a GEC. Thus, we can restrict attention to maximal GEC. For a GEC C, there exists a memoryless strategy f^C that maximizes the probability of reaching $C^{\max}$ from every state $s \in C$ while staying in C. Moreover, since C is an EC, the probability of reaching $C^{\max}$ by playing f^C is strictly positive from every state $s \in C$. Let t be a state in C. Consider the MDP $\mathcal{M}^{\mathtt{MDP}}|_C^t$. Since C is EC, we have that $cost(\mathcal{M}^{\mathtt{MDP}}|_C^t)$ is independent of the initial state t. Thus, we can define $cost(\mathcal{M}^{\mathtt{MDP}}|_C)$ as $cost(\mathcal{M}^{\mathtt{MDP}}|_C^t)$ for some $t \in C$.

Recall that our algorithm starts by a preprocessing step that removes all states that are not sure-winning for Player 1 in $\mathcal{M}^{\mathtt{P}}$. It then finds the maximal GECs of $\mathcal{M}$ (using a polynomial-time procedure that we describe in Appendix A.1), and obtain an MDP $\mathcal{M}'$ by assigning every state within a GEC C the cost $cost(\mathcal{M}^{\mathtt{MDP}}|_C)$, and assigning every state that is not inside a GEC cost $W+1$, where W is the maximal cost that appears in $\mathcal{M}$. We claim that $cost_{\mathrm{sure}}(\mathcal{M}) = cost(\mathcal{M}')$.

Before proving the claim, note that all the steps of the algorithm except for the preprocessing step that involves a solution of parity game require polynomial time. In particular, the strategies f^C above are computable in polynomial time by solving a reachability MDP, and, by Theorem 1, so does the final step of finding $cost(M')$.

Proving that $cost_{\mathrm{sure}}(\mathcal{M}) = cost(\mathcal{M}')$ involves the following steps (see Appendix A for the full proof). First, proving $cost_{\mathrm{sure}}(\mathcal{M}) \geq cost(\mathcal{M}')$ is not hard, as a play with a winning strategy f for Player 1 in $\mathcal{M}$ reaches and stays in some GEC C w.p. 1, and within C, the best expected cost one can hope for is $cost(\mathcal{M}^{\mathtt{MDP}}|_C)$, which is exactly what the strategy f attains when played in $\mathcal{M}'$.

Next, proving $cost_{\mathrm{sure}}(\mathcal{M}) \leq cost(\mathcal{M}')$, we show how an optimal strategy f' in $\mathcal{M}'$ induces an ϵ-optimal strategy f in $\mathcal{M}$. We start with Lemma 1, which justifies the costs within a GEC.

Lemma 1. *Consider a GEC C in $\mathcal{M}$, and $s \in C$. Let $v(s) = cost(\mathcal{M}^{\mathtt{MDP}}|_C^s)$, then for every $\epsilon > 0$ there exists a strategy f of $\mathcal{M}^s$ with $cost_{\mathrm{sure}}(f) \leq v(s) + \epsilon$.*

Intuitively, in a good EC, f minimizes the expected mean-payoff and *once in a while* it plays reachability, aiming to visit to a state with the maximal rank in the EC. Since the EC is good, this rank is even. If reachability is not obtained after N rounds, for a parameter N, then f gives up and aims at only surely satisfying the parity objective (our preprocessing step ensures that this is possible). Otherwise, after reaching the maximal rank, f switches again to minimizing mean-payoff. This process is repeated forever, increasing N in each iteration. Hence, the probability that Player 1 eventually gives up can be bounded from above by an arbitrarily small $\epsilon > 0$. Accordingly, Player 1 can achieve a value that is arbitrarily close to $cost(\mathcal{M}^{\mathtt{MDP}}|_C)$.

Finally, we construct the ϵ-optimal strategy f in $\mathcal{M}$ as follows. The strategy f first mimics f' for a large number of steps k, or until an EC (in which f' stays forever) is reached. If a good EC is not reached, then f aims at only surely satisfying the parity objective. If a good EC is reached, then f behaves as prescribed above, per Lemma 1. Since the probability of f' reaching a good EC within k steps tends to 1, then Player 1 can achieve a value within ϵ of $cost(\mathcal{M}')$.

3.2 Finite-Memory Strategies

In this section we study the problem of finding the sure cost of a parity-MDP, when restricted to finite memory strategies. For a parity-MDP $\mathcal{M}$, we define $cost_{\mathrm{sure},<\infty}(\mathcal{M}) = \inf\{cost_{\mathrm{sure}}(f) : f \text{ is a finite memory strategy for } \mathcal{M}\}$. We prove the upper bound in the following theorem. As stated above, the lower bound is trivial.

Theorem 4. *Consider a parity-MDP $\mathcal{M}$. Then, $cost_{\mathrm{sure},<\infty}(\mathcal{M})$ can be computed in NP$\cap$co-NP, and is parity-games hard.*

The general approach is similar to the one we took in Section 3.1. That is, we remove from $\mathcal{M}$ all states that are not sure-winning for Player 1 in $\mathcal{M}^{\mathsf{P}}$, and proceed by reasoning about a certain type of ECs. However, for finite-memory strategies, we need a more restricted class of ECs than the GECs that were used in Section 3.1. Indeed, a finite-memory strategy might not suffice to win the sure-parity condition in a GEC.

For a GEC C, let k be the maximal odd priority in C, with $k = -1$ if there are no odd priorities. We define $C_{\mathrm{even}}^{\max} = \{q \in C : \alpha(q) > k \text{ and } \alpha(q) \text{ is even}\}$. We say that a GEC C in $\mathcal{M}$ is *super good* (SGEC, for short) if from every state $s \in C$, there exists a finite-memory strategy f for $\mathcal{M}|_C^s$ such that the play of $\mathcal{M}$ under f reaches $C_{\mathrm{even}}^{\max}$ w.p. 1, and if the play does not reach $C_{\mathrm{even}}^{\max}$, then it is parity winning. We refer to f as a *witness* to C being a SGEC. If C is not a SGEC, we refer to the states of C that satisfy the above as *super-good states*.

We argue that SGECs are the proper notion for reasoning about finite-memory strategies. Specifically, we show that in a SGEC, Player 1 can achieve ϵ-optimal expected cost with a finite-memory strategy, and that every finite-memory winning strategy reaches a SGEC w.p. 1.

Our algorithm finds the maximal SGECs of $\mathcal{M}$ and obtain an MDP $\mathcal{M}'$ in the same manner we did in Section 3.1, namely by assigning high weights to states not in SGECs, and the optimal mean-payoff MDP value to states in SGECs. As there, we claim that

$cost(\mathcal{M}') = cost_{\text{sure},<\infty}(\mathcal{M})$. The analysis of the algorithm as well as its concrete details, are, however, more intricate.

We start by proving that the notion of maximal SGECs is well defined. To this end, we present the following lemma, whose proof appears in Appendix B.1. Note that in the case of GECs, the lemma was trivial.

Lemma 2. *Consider SGEC C and D, such that $C \cap D \neq \emptyset$, then $C \cup D$ is also a SGEC.*

Intuitively, we prove this by considering witnesses f, g for C and D being SGECs. We then modify f such that from every state in C, it tries to reach D for N steps, for some parameter N. Once D is reached, g takes over. If D is not reached, f attempts to reach $C_{\text{even}}^{\text{max}}$. Thus, w.p. 1, the strategy reaches $D_{\text{even}}^{\text{max}}$, and if it does not, it either reaches $C_{\text{even}}^{\text{max}}$ infinitely often, or wins the parity condition.

Next, we note that unlike the syntactic definition of GECs, the definition of SGECs is semantic, as it involves a strategy. Thus, finding the maximal SGECs adds another complication to the algorithm. In fact, it is not hard to see that even checking whether an EC is a SGEC is parity-games hard. Using techniques from [7], we show in Appendix B.2 that we can reduce the latter to the problem of solving a parity-Büchi game. We thus have the following lemma.

Lemma 3. *Consider an EC C in a parity-MDP $\mathcal{M}$. We can decide whether C is a SGEC in NP$\cap$ co-NP, as well as compute a witness strategy and, if C is not a SGEC, find the set of super-good states.*

Next, we show how to find the maximal SGECs of $\mathcal{M}$. Essentially, for every odd rank k, we can find the SGECs whose maximal odd rank is k by removing all states with higher odd ranks, and recursively refining ECs by keeping only super-good states, using Lemma 3. Thus, we have the following (see Appendix B.3 for complete details).

Theorem 5. *Consider a parity-MDP $\mathcal{M}$. We can find the maximal SGECs of $\mathcal{M}$ in NP$\cap$co-NP.*

Theorem 5 shows that our algorithm for computing $cost_{\text{sure},<\infty}(\mathcal{M})$ solves the problem in NP$\cap$co-NP. It remains to prove its correctness. First, Lemma 4 justifies the assignment of costs within a SGEC.

Lemma 4. *Consider a SGEC C in $\mathcal{M}$ and a state s in C. Let $v(s) = cost(\mathcal{M}^{\texttt{MDP}}|_C^s)$. Then, for every $\epsilon > 0$, there exists a finite-memory strategy f of $\mathcal{M}|_C^s$ with $cost_{\text{sure}}(f) \leq v(s) + \epsilon$.*

Proof. Let g be a memoryless strategy such that $cost(g) = cost(\mathcal{M}^{\texttt{MDP}}|_C^s)$. By Theorem 1 such a strategy exists. Let h be a finite-memory strategy that witnesses C being a SGEC. For every $k \in \mathbb{N}$, consider the strategy f_k that repeatedly plays g for k steps and then plays h until $C_{\text{even}}^{\text{max}}$ is reached. Since g and h are finite-memory, then f_k is finite memory. In addition, observe that h reaches $C_{\text{even}}^{\text{max}}$ w.p. 1, and if $C_{\text{even}}^{\text{max}}$ is not reached, then h is parity-winning. Thus f_k is parity-winning, and it reaches Step 1 infinitely often w.p. 1. Moreover, since h has finite memory, then for every $n \in \mathbb{N}$, there is a bounded probability $0 < p(n) \leq 1$ that f reaches $C_{\text{even}}^{\text{max}}$ within n steps, with $\lim_{n\to\infty} p(n) = 1$. Thus, we get that $\lim_{k\to\infty} scost(f_k) = scost(g) = cost(\mathcal{M}^{\texttt{MDP}}|_C^s)$, which concludes the proof.

Lemma 4 implies that we can approximate the optimal value of SGECs with finite-memory strategies. It remains to show that it is indeed enough to consider SGECs. Consider a finite-memory strategy f. Then, w.p. 1, f reaches an EC. Let C be an EC with $\Pr_{\mathcal{M}}(\text{inf}(f) = C) > 0$. The following lemma characterizes an assumption we can make on the behavior of f in such an EC.

Lemma 5. *Consider a parity-MDP $\mathcal{M}$ and an EC C. For every finite-memory strategy f, if $\Pr_{\mathcal{M}}(inf(f) = C) > 0$, then there exists a finite-memory strategy g such that for every $s \in C$, we have that $\Pr_{\mathcal{M}^s}(inf(g) = C) = 1$ and every play of g from s stays in C. Moreover, if f is parity winning, then so is g.*

Intuitively, we show that there exists some finite history h such that the strategy f_h, which is f played after seeing the history h, has the following property: f_h reaches and stays in C, and w.p. 1 visits infinitely often all the states in C, and in particular $C^{\max}_{\text{even}}$. For the proof, we consider the set $F = \{f_h : h \text{ is a finite history}\}$. Since f has finite memory, it follows that this set is finite. Using this, we show that if $\Pr_{\mathcal{M}}(\inf(g) = C) < 1$ for every $g \in F$, then $\Pr_{\mathcal{M}}(\inf(f) = C) = 0$, which is a contradiction. Finally, since f is also parity winning, it follows that f_h above is also parity-winning, and is thus a witness for C being a SGEC. The full proof appears in Appendix B.4.

Finally, by Lemma 6, we can assume that once f reaches an EC C, it stays in C and visits all its states infinitely often w.p. 1. Since f is parity-winning, it follows that C has a maximal even rank, and that f reaches $C^{\max}_{\text{even}}$ w.p. 1. Moreover, in every play that does not reach $C^{\max}_{\text{even}}$, f wins the parity condition. We can thus conclude with the following Lemma, which completes the correctness proof of our algorithm for computing $cost_{\text{sure},<\infty}(\mathcal{M})$. See Appendix B.5 for the proof.

Lemma 6. *Consider a parity-MDP $\mathcal{M}$ and an EC C. For every finite-memory strategy f, if f is parity winning and $\Pr_{\mathcal{M}}(inf(f) = C) > 0$, then C is a SGEC.*

3.3 Comparison with Related Work

Both our work and [6,12] analyze ECs and reduce the problem to reasoning about an MDP that ignores the hard constraints. The main difference with [6] is that there, the hard and soft constraints have the same objective (i.e., worst-case mean-payoff value and expected-case mean-payoff value). In [6], the strategy played for N rounds to satisfy the soft objective and then at most M rounds to satisfy the hard objective, for some constants N and M. In our setting, we cannot bound M, and in fact it might be the case that Player 1 would play to satisfy the parity objective for the rest of the game (i.e., forever) even after reaching a super-good end component.

The difference with [12] is twofold. First, technically, the type of hard constraints in [12] is worst-case mean-payoff, whereas our setting uses the Boolean parity condition. In classical parity games, the parity condition can be reduced to a mean-payoff objective. Similar reductions, however, do not work in order to reduce our setting to the setting of [12]. Thus, our contribution is orthogonal to [12]. Secondly, Boolean constraints are conceptually different than quantitative constraints, and as we demonstrate in Section 4, they arise naturally in quantitative extensions of Boolean paradigms.

We note that [12] also study a relaxation in which almost-sure winning is allowed for the hard constraints. An analogue in our setting is to consider an almost-sure parity condition. We note that in such a setting, GECs are sufficient for reasoning both about finite-memory and infinite-memory strategies. Moreover, the preprocessing involves solving an almost-sure parity MDP (without mean-payoff constraints), which can be done in polynomial time. Thus, as is the case in [12], we can compute the cost of an MDP with almost-sure hard constraints in polynomial time.

4 Applications

In this section we study two applications of parity-MDPs. Both extend the Boolean synthesis problem. Due to lack of space, our description is only an overview. Full definitions and details can be found in Appendix C. We start with some basic definitions.

For finite sets I and O of input and output signals, respectively, an I/O *transducer* is $\mathcal{T} = \langle I, O, Q, q_0, \delta, \rho \rangle$, where Q is a set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times 2^I \to Q$ is a total (deterministic) transition function, and $\rho : Q \to 2^O$ is a labeling function on the states. The run of $\mathcal{T}$ on a word $w = i_0 \cdot i_1 \cdots \in (2^I)^\omega$ is the sequence of states $q_0, q_1, \ldots$ such that $q_{k+1} = \delta(q_k, i_k)$ for all $k \geq 0$. The *output* of $\mathcal{T}$ on w is then $o_1, o_2, \ldots \in (2^O)^\omega$ where $o_k = \rho(q_k)$ for all $k \geq 1$. Note that the first output assignment is that of q_1, and we do not consider $\rho(q_0)$. This reflects the fact that the environment initiates the interaction. The *computation of $\mathcal{T}$ on w* is then $\mathcal{T}(w) = i_0 \cup o_1, i_1 \cup o_2, \ldots \in (2^{I \cup O})^\omega$. When Q is a finite set, we say that the transducer is finite.

The synthesis problem gets as input a specification $L \subseteq (2^{I \cup O})^\omega$ and generates a transducer $\mathcal{T}$ that realizes L; namely, all the computations of $\mathcal{T}$ are in L. The language L is typically given by an LTL formula [17] or by means of an automaton of infinite words.

4.1 Penalties on Undesired Scenarios

Recall that in the Boolean synthesis problem, the goal is to generate a transducer that associates with each infinite sequence of inputs an infinite sequence of outputs so that the result computation satisfies a given specification. Typically, some behaviors generated by the transducers may be less desired than others. For example, as discussed in Section 1, designs that use fewer resources or minimize expensive activities are preferable. The input to the *synthesis with penalties* problem includes, in addition to the Boolean specification, languages of finite words that describe undesired behaviors, and their costs. The goal is to generate a transducer that realizes the specification and minimizes cost due to undesired behaviors.

Formally, the input to the problem includes languages $L_1, \ldots, L_m$ of finite words over the alphabet $2^{I \cup O}$ and a penalty function $\gamma : \{1, \ldots, m\} \to \mathbb{N}$ specifying for each $1 \leq i \leq m$ the penalty that should be applied for generating a behavior in L_i. As described in Section 1, the language L_i may be local (that is, include only words of length 1) and thus refer only to activation of output signals, may specify short scenarios like flips of output signals, and may also specify rich regular scenarios. Note that we allow penalties also for behaviors that depend on the input signals. Intuitively, whenever a computation π includes a behavior in L_i, a penalty of $\gamma(i)$ is applied. Formally, if $\pi = \sigma_1, \sigma_2, \ldots$, then for every position $j \geq 1$, we define $penalty(j) = \{i :$ there is $k \leq i$ such that $\sigma_k \cdot \sigma_{k+1} \cdots \sigma_j \in L_i\}$. That is, $penalty(j)$ points to the subset of languages L_i such that a word in L_i ends in position j. Then, the cost of position j, denoted $cost(j)$, is $\sum_{i \in penalty(j)} \gamma(i)$. Finally, for a finite computation $\pi = \sigma_1, \sigma_2, \ldots$, we define its cost, denoted $cost(\pi)$, as $\limsup_{m \to \infty} \frac{1}{m} \sum_{j=1}^{m} cost(j)$.

Let $\mathcal{A}$ be a deterministic parity automaton (DPW, for short) over the alphabet $2^{I \cup O}$ that specifies the specification ψ. We describe a parity-MDP whose solution is a transducer that realizes $\mathcal{A}$ with the minimal cost for penalties. The idea is simple: on top of the parity game $\mathcal{G}$ described above, we compose monitors that detect undesired scenarios. We assume that the languages $L_1, \ldots, L_m$ and are given by means of deterministic automata on finite words (DFWs) $\mathcal{U}_1, \ldots, \mathcal{U}_m$ where for every $1 \leq i \leq m$, we have that $L(\mathcal{U}_i) = (2^{I \cup O})^* \cdot L_i$. That is, $\mathcal{U}_i$ accepts $\sigma_1 \cdots \sigma_n$ iff there exists $k \leq n$ such that $\sigma_k \cdots \sigma_n \in L_i$. Essentially, we turn $\mathcal{A}$ into a parity-MDP by composing it with the DFWs $\mathcal{U}_1, \ldots, \mathcal{U}_m$. Then, $\mathcal{U}_i$ reaching an accepting state indicates that the penalty for L_i should be applied, which induces the costs in the parity-MDP. The probabilities in the parity-MDP are induced form the distribution of the assignments to input signals. The full construction can be found in Appendix C.2. We note that an alternative definition can replace the DFWs $\mathcal{U}_1, \ldots, \mathcal{U}_m$ and the cost function γ by a single weighted automaton that can be composed with $\mathcal{A}$.

4.2 Sensing

Consider a transducer $\mathcal{T} = \langle I, O, Q, q_0, \delta, \rho \rangle$. For a state $q \in Q$ and a signal $p \in I$, we say that p is *sensed in* q if there exists a set $S \subseteq I$ such that $\delta(q, S \setminus \{p\}) \neq \delta(q, S \cup \{p\})$. Intuitively, a signal is sensed in q if knowing its value may affect the destination of at least one transition from q. We use $sensed(q)$ to denote the set of signals sensed in q. The *sensing cost* of a state $q \in Q$ is $scost(q) = |sensed(q)|$. For a finite run $r = q_1, \ldots, q_m$ of $\mathcal{T}$, we define the sensing cost of r, denoted $scost(r)$, as $\frac{1}{m} \sum_{i=0}^{m-1} scost(q_i)$. That is, $scost(r)$ is the average number of sensors that $\mathcal{T}$ uses during r. For a finite input sequence $w \in (2^I)^*$, we define the sensing cost of w in $\mathcal{T}$, denoted $scost_{\mathcal{T}}(w)$, as the sensing cost of the run of $\mathcal{T}$ on w. Finally, the sensing cost of $\mathcal{T}$ is the expected sensing cost of input sequences of length that tends to infinity, which is parameterized by a distribution on $(2^I)^\omega$ given by a sequence of distributions $D_1, D_2, ...$ such that $D_t : 2^I \to [0,1]$ describes the distribution over 2^I at time $t \in \mathbb{N}$. For simplicity, we assume that the distribution is uniform. Thus, $D_t(i) = 2^{-|I|}$ for every $t \in \mathbb{N}$. For the uniform distribution we have $scost(\mathcal{T}) = \lim_{m \to \infty} |(2^I)|^{-m} \sum_{w \in (2^I)^m} scost_{\mathcal{T}}(w)$.

Note that this definition also applies when the transducer is infinite. However, for infinite transducers, the limit in the definition of $scost(\mathcal{T})$ might not exist, and we therefore define $scost(\mathcal{T}) = \limsup_{m \to \infty} |2^I|^{-m} \sum_{w \in (2^I)^m} scost_{\mathcal{T}}(w)$. Finally, for a realizable specification $L \in 2^{I \cup O}$, we define $scost_{I/O}(L) = \inf\{scost(\mathcal{T}) : \mathcal{T}$ is an I/O transducer that realizes $L\}$.

In [3], we study the sensing cost of *safety* properties. We show that there, a finite, minimally-sensing transducer, always exists (albeit of exponential size), and the problem of computing the sensing cost is EXPTIME-complete. In our current setting, however, a minimally-sensing transducer need not exist, and any approximation may require infinite memory. We demonstrate this with an example.

Example 6 Let $I = \{a\}$ and $O = \{b\}$, and consider the specification $\psi = (\mathsf{GF}a \wedge \mathsf{G}b) \vee \mathsf{G}(\neg b \to \mathsf{XG}(a \leftrightarrow b))$. Thus, ψ states that either a holds infinitely often and b always holds, or, if b does not holds at a certain time, then henceforth, a holds iff b holds. Observe that once the system outputs $\neg b$, it has to always sense a in order to determine the output. The system thus has an incentive to always output b. This, however, may render ψ false, as a need not hold infinitely often.

We start by claiming that every finite-memory transducer $\mathcal{T}$ that realizes ψ has sensing cost 1. Indeed, let n be the number of states in $\mathcal{T}$. A random input sequence contains the infix $(\neg a)^{n+1}$ w.p. 1. Upon reading such an infix, $\mathcal{T}$ has to output $\neg b$, as otherwise it would not realize ψ on a computation with suffix $(\neg a)^\omega$. Thus, from then on, $\mathcal{T}$ senses a in every state. So $scost(\mathcal{T}) = 1$.

However, by using infinite-memory transducers, we can follow the construction in Section 3.1 and reduce the sensing cost arbitrarily close to 0. Let $M \in \mathbb{N}$. We construct a transducer $\mathcal{T}'$ as follows. After initializing i to 1, the transducer $\mathcal{T}'$ senses a and outputs b for iM steps. If a does not hold during this time, then $\mathcal{T}'$ outputs $\neg b$ and starts sensing a and outputting b accordingly. Otherwise, if a holds during this time, then $\mathcal{T}'$ stops sensing a for 2^i steps, while outputting b. It then increases i by 1 and repeats the process. Note that $\mathcal{T}'$ outputs $\neg b$ iff a does not hold for iM consecutive positions at the i-th round (which happens w.p. 2^{-iM}). Thus, the probability of $\mathcal{T}'$ outputting $\neg b$ in a random computation is bounded from above by $\sum_{i=1}^{\infty} 2^{-iM} = 2^{-M}$, which tends to 0 as M tends to ∞. Note that in the i-th round, $\mathcal{T}'$ senses a for only iM steps, and then does not sense anything for 2^{iM} steps, so if $\mathcal{T}'$ does not output $\neg b$, the sensing cost is 0. Thus, we have $\lim_{M\to\infty} scost(\mathcal{T}') = 0$. □

We proceed by describing the general solution to computing the sensing cost of a specification. Recall that synthesis of a DPW $\mathcal{A}$ is reduced to solving a parity game. When sensing is introduced, it is not enough for the system to win this game, as it now has to win while minimizing the sensing cost. Intuitively, not sensing some inputs introduces incomplete information to the game: once the system gives up sensing, it may not know the state in which the game is and knows instead only a set of states in which the game may be. In particular, unlike usual realizability, a strategy that minimizes the sensing need not use the state space of the DPW.

Theorem 7. *Consider a* DPW *specification $\mathcal{A}$ over $2^{I\cup O}$. There exists a parity-MDP $\mathcal{M}$ such that $cost_{\text{sure}}(\mathcal{M}) = scost_{I/O}(L(\mathcal{A}))$. Moreover, the number of states of $\mathcal{M}$ is singly exponential in that of $\mathcal{A}$, and the number of parity ranks on $\mathcal{M}$ is polynomial in that of $\mathcal{A}$.*

Proof. Conceptually, we follow the ideas of Boolean synthesis, by thinking of $\mathcal{A}$ as a parity game between the system and the environment, as described in Section C.1. The proof is comprised of several steps. First, intuitively, we give the system an option to sense only some input signals $x \subseteq I$, but require that then, the play must be winning for every assignment of the inputs that are not sensed. Then, we introduce costs induced by the number of sensed input signals in each state, and finally we add a uniform stochastic environment. Technically, however, the first step is done using automata, rather than games, and converts the DPW $\mathcal{A}$ into a *universal parity automaton* (UPW) – an automaton in which a the transition function maps each state and letter to more than a single successor state, and a word is accepted if all the runs on it are accepting. We use the universal branches of the UPW in order to model the several possible assignments to the input signals that are not sensed. Thus, in state s of $\mathcal{A}$, the system chooses a state $\langle s, x\rangle$, where $x \subseteq I$ represents the inputs to be sensed. The environment then chooses an assignment $i : I \to \{0, 1\}$ for the inputs, and the system chooses an output assignment $o : O \to \{0, 1\}$. However, instead of the new state being $\delta(s, i \cup o)$, a universal transition is taken to every state s' such that $s' = \delta(s, i' \cup o)$ for some i' that agrees with i on every input in x. Thus, effectively, the system has to play only according to the values of the sensed inputs. Note that the two players are not modeled in the automaton. Rather, their choices are represented by augmenting the alphabet to include a 2^I component to represent the sensed inputs. Using automata allows us to determinize the UPW back to a DPW that already captures sensing. We then convert the automaton to a parity-game, and proceed as described above.

For the formal details, see Appendix C.3.

Theorem 8. *Consider a* DPW *specification* $\mathcal{A}$ *over* $2^{I \cup O}$*. We can compute* $scost_{I/O}(L(\mathcal{A}))$ *in singly-exponential time. Moreover, the problem of deciding whether* $scost_{I/O}(L(\mathcal{A})) > 0$ *is EXPTIME-complete.*

Proof. We obtain from $\mathcal{A}$ a parity-MDP $\mathcal{M}$ as per Theorem 7. Observe that the algorithm in the proof of Theorem 3 essentially runs in polynomial time, apart from solving a parity game, which is done in NP∩co-NP. However, deterministic algorithms for solving parity games run in time polynomial in the number of states, and singly-exponential in the number of parity ranks. Since the number of parity ranks in $\mathcal{M}$ is polynomial in that of $\mathcal{A}$, we can find $cost_{\rm sure}(\mathcal{M})$ in time singly-exponential in the size of $\mathcal{A}$. Since $cost_{\rm sure}(\mathcal{M}) = scost_{I/O}(L(\mathcal{A}))$, we are done.

For the lower bound, we note that the problem of deciding whether $scost_{I/O}(L(\mathcal{A})) > 0$ is EXPTIME-hard even for a restricted class of automata, namely looping automata [3].

The input to the synthesis problem is typically given as an LTL formula, rather than a DPW. Then, the translation from LTL to a DPW involves a doubly-exponential blowup. Thus, a naive solution for computing the sensing cost of a specification given by an LTL formula is in 3EXPTIME. However, by translating the formula to a UPW, rather than a DPW, we show how we can avoid one exponent, thus matching the 2EXPTIME complexity of standard Boolean synthesis.

Theorem 9. *Consider an* LTL *specification* ψ *over* $I \cup O$*. We can compute* $scost_{I/O}(L(\psi))$ *in doubly-exponential time.*

Proof. We start by translating ψ to a UPW $\mathcal{A}$ of size single-exponential in the size of ψ. This can be done, for example, by translating $\neg\psi$ to a nondeterministic Büchi automaton [21] and dualizing it. We then follow the proof of Theorem 8, by adding the universal transitions described there directly to the UPW $\mathcal{A}$. Thus, when we finally determinize the UPW to a DPW, the size of the DPW is doubly-exponential, so computing the sensing cost can also be done in doubly-exponential time.

References

1. S. Almagor, U. Boker, and O. Kupferman. Formalizing and reasoning about quality. *Journal of the ACM*, 2016. To appear.
2. S. Almagor, D. Kuperberg, and O. Kupferman. Regular sensing. In *Proc. 34th FSTTCS*, *LIPIcs* 29, pages 161–173, 2014.
3. S. Almagor, D. Kuperberg, and O. Kupferman. The sensing cost of monitoring and synthesis. In *Proc. 35th FSTTCS*, *LIPIcs* 35, pages 380–393, 2015.
4. E. Arbel, O. Rokhlenko, and K. Yorav. Sat-based synthesis of clock gating functions using 3-valued abstraction. In *FMCAD*, pages 198–204, 2009.

5. P. Bouyer, N. Markey, and R. Matteplackel. Averaging in LTL. In *Proc. 25th CONCUR*, pages 266–280, 2014.
6. V. Bruyère, E. Filiot, M. Randour, and J-F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. In *Proc. 31th STACS, LIPIcs* 25, pages 199–213, 2014.
7. K. Chatterjee and L. Doyen. Energy and mean-payoff parity markov decision processes. In *36th MFCS*, pages 206–218, 2011.
8. K. Chatterjee, L. Doyen, H. Gimbert, and Y. Oualhadj. Perfect-information stochastic mean-payoff parity games. In *Proc. 17th FOSSACS*, pages 210–225, 2014.
9. K. Chatterjee and M. Henzinger. Efficient and dynamic algorithms for alternating büchi games and maximal end-component decomposition. *Journal of the ACM*, 61(3):15:1–15:40, 2014.
10. K. Chatterjee, Z. Komárková, and J. Kretínský. Unifying two views on multiple mean-payoff objectives in markov decision processes. In *Proc. 30th LICS*, pages 244–256, 2015.
11. A. Church. Logic, arithmetics, and automata. In *Proc. Int. Congress of Mathematicians, 1963*, 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 *Proc. 30th LICS*, pages 257–268, 2015.
13. L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. *Theoretical Computer Science*, 345(1):139–170, 2005.
14. C. Eisner, A. Nahir, and K. Yorav. Functional verification of power gated designs by compositional reasoning. *Formal Methods in System Design*, 35(1):40–55, 2009.
15. M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi. *Low Power Methodology Manual*. Springer, 2007.
16. O. Kupferman and M.Y. Vardi. Synthesis with incomplete information. In *Advances in Temporal Logic*, pages 109–127. Kluwer Academic Publishers, 2000.
17. A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.
18. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th POPL*, pages 179–190, 1989.
19. M.L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
20. R. Rosner. *Modular Synthesis of Reactive Systems*. PhD thesis, Weizmann Institute of Science, 1992.
21. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *I&C*, 115(1):1–37, 1994.

A Calculating the Sure Cost in the Infinite-Memory Case

Our algorithm uses the notion of attractors, defined below. Consider a set $R \subseteq S$. A *environment attractor* for R, denoted $\mathrm{Attr}_{\mathrm{env}}(R)$ is defined inductively as follows. First, $T_0 = R$. Now, for every $i > 0$, let $T_{i+1} = T_i \cup \{s \in S_1 : \text{ for every } a \in A_1(s), \text{ we have that } \delta_1(s,a) \in T_i\} \cup \{s \in S_2 : \text{ there exists } a \in A_2(s) \text{ s.t. } \delta_2(s,a) \in T_i$ and P(s,a)¿0$\}$. Then, $\mathrm{Attr}_{\mathrm{env}}(R) = \bigcup_i T_i$. It is well known that $\mathrm{Attr}_{\mathrm{env}}(R)$ can be computed in time polynomial in the description of $\mathcal{M}$. We analogously define the *system attractor* $\mathrm{Attr}_{\mathrm{sys}}(R)$, by swapping the roles of Players 1 and 2.

A.1 Finding the Maximal GECs of $\mathcal{M}$

In order to find the maximal GECs of $\mathcal{M}$, we proceed as follows.

1. Compute the maximal EC decomposition of $\mathcal{M}$.
2. For every maximal EC C, if C is not good (i.e., the maximal parity rank in C is odd), remove it from the graph $\mathrm{Attr}_{\mathrm{env}}(C^{\max})$ and go to (1).
3. Once all the remaining components are good, return them.

Note that upon returning to step (1) from (2), it may be that the graph of $\mathcal{M}$ is not connected. Still, we find the decomposition in all the components.

It is not hard to see that all the steps of the algorithm are polynimial. In particular, finding the maximal EC decomposition of $\mathcal{M}$ takes polynomial time [9].

A.2 Proof of Lemma 1

Consider a memoryless strategy f^C that maximizes the probability to reach $C^{\max}$, and a memoryless strategy g whose expected cost in $\mathcal{M}^{\mathtt{MDP}}|_C^s$ is $v(s) = v(C)$. By Theorem 1, such a strategy g exists.

We construct an infinite-memory strategy h that works in phases, as follows. In phase 1, h works in iterations. In iteration i, the strategy h plays g for 2^{2^i} steps. Then, h plays f^C for $\gamma_\epsilon \cdot n \cdot 2^i$ steps, where γ_ϵ is a constant we determine later and n is the number of states of $\mathcal{M}$. If, during these $2^{2^i} + \gamma_\epsilon 2^i$ steps, the generated play reached $C^{\max}$, then we proceed to the next iteration. Otherwise, h goes to phase 2, in which it plays a parity-winning strategy (which exists, since every state in $\mathcal{M}$ is parity winning).

Clearly, if the play generated by h never reaches Phase 2, then playing g for 2^{2^i} steps is the dominant factor, and we have that $cost(h) = cost(g) \leq v(s)$. Thus, it remains to bound the probability that the play reaches Phase 2. Denote by λ the maximal probability that a play of f^C does not reach $C^{\max}$ within n steps, where the maximum is taken over all states of C. Since C is strongly connected, it follows that $0 \leq \lambda < 1$. Thus, the probability of not reaching Phase 2 is bounded from below by $\prod_{i=1}^{\infty}(1 - \lambda^{\gamma_\epsilon 2^i})$. The latter expression converges to a number p in $(0, 1]$ that is inversely-related to γ_ϵ. Therefore, by setting γ_ϵ large enough, we can lower the probability of reaching Phase 2 arbitrarily. Since the cost of a play after reaching Phase 2 is bounded from above by W, the claim follows.

A.3 A proof that $cost_{\mathrm{sure}}(\mathcal{M}) = cost(\mathcal{M}')$

We start with the "easy" direction, proving that $cost_{\mathrm{sure}}(\mathcal{M}) \geq cost(M')$. Consider a winning strategy f for $\mathcal{M}$. With probability 1, the play of f in $\mathcal{M}$ reaches and stays in some GEC C. From every state in C, the minimal expected cost (when staying in C) is $v(C)$. Indeed, $v(C)$ is the cost of an MDP without the parity condition, which can only lower the minimal expected cost. Thus, we have that $cost_{\mathcal{M}}(f) \geq \sum_{C \text{ is a GEC}} \Pr(f \text{ reaches and stays in } C) \cdot v(C)$.

Consider the strategy f as a strategy for $\mathcal{M}'$. Then, $cost_{\mathcal{M}'}(f) = \sum_{C \text{ is a GEC}} \Pr(f \text{ reaches and stays in } C) \cdot v(C)$, and we conclude that $cost_{\mathrm{sure}}(\mathcal{M}) \geq cost(\mathcal{M}')$.

For the other direction, we show that $cost_{\mathrm{sure}}(\mathcal{M}) \leq cost(\mathcal{M}')$. Since $\mathcal{M}'$ is an MDP, then there exists an optimal memoryless strategy f' such that $cost_{\mathcal{M}'}(f') = cost(\mathcal{M}')$. We show that for every $\epsilon > 0$, there exists a winning strategy f for $\mathcal{M}$ such that $cost_{\mathcal{M}}(f) \leq cost_{\mathcal{M}'}(f') + \epsilon$.

Observe that since f' is memoryless and optimal, there exists a set of ECs $\mathcal{C}$ such that for every $C \in \mathcal{C}$, once f' reaches a state $s \in C$, it stays in C forever. Moreover, observe that every $C \in \mathcal{C}$ must be a GEC. Indeed, the states outside a GEC in $\mathcal{M}'$ have value $2W + 1$, but from every state in $\mathcal{M}$ there exists a strategy that is parity-winning, and therefore ensures that a GEC is reached. Thus, if f' gets stuck in an EC that is not good, we can modify it to reach a GEC, thus decreasing its cost.

Let $\epsilon > 0$. There exists some $N_0 \in \mathbb{N}$ such that after N_0 steps, w.p. at least $1 - \epsilon'$ a play in $\mathcal{M}'_{f'}$ reaches a GEC in $\mathcal{C}$ (for $\epsilon' > 0$ which we will fix later). We obtain f from f' as follows. f simulates f' for N_0 steps. During this simulation, whenever f reaches a GEC $C \in \mathcal{C}$, f starts playing the strategy described in the proof of Lemma 1 for ϵ'. After N_0 steps, f plays a parity-winning strategy.

Clearly f is parity-winning. In addition, by our choice of N_0 and by Lemma 1, it follows that w.p. at least $1-\epsilon'$, the cost of f is at most $cost_{\mathcal{M}'}(f)+\epsilon'$. Thus, $cost_{\mathcal{M}}(f) \leq (1-\epsilon')(cost_{\mathcal{M}'}(f)+\epsilon')+\epsilon'|W|$, and for a small enough ϵ', this is at most $cost_{\mathcal{M}'}(f)+\epsilon$.

B Calculating the Sure Cost in the finite-Memory Case

B.1 Proof of Lemma 2

Assume w.l.o.g that the maximal odd priority in C is at least that of D. Let f, g be witnesses to C and D being SGEC, respectively. We construct a witness h to $C \cup D$ being a SGEC. In every state $q \in C$, h behaves as f does. In a state $s \in D \setminus C$, h proceeds as follows. (1) It attempts to reach a state $s' \in C \cap D$ (from which it behaves as f) within $n_0 \in \mathbb{N}$ steps, for a large enough n_0 such that the probability of reaching C is positive (which exists, since $C \cup D$ is an EC). (2) If C was not reached within n_0 steps, h plays g until $C^{\max}_{\text{even}}$ is reached, and goes back to (1).

Observe that $C^{\max}_{\text{even}} \subseteq (C \cup D)^{\max}_{\text{even}}$. Clearly, the play under h reaches $C^{\max}_{\text{even}}$ w.p. 1. Moreover, if the play does not reach $C^{\max}_{\text{even}}$, then it is winning in the parity condition. Indeed, if the play under h reaches C, then this holds (since f is a witness for C being a SGEC). Otherwise, the play of h either reaches $D^{\max}_{\text{even}}$ infinitely often, in which case it is winning in the parity condition, or it plays as g and does not reach $D^{\max}_{\text{even}}$, in which case it is parity winning, since g is a witness for D being a SGEC. We conclude that $C \cup D$ is a SGEC.

B.2 Proof of Lemma 3

Our solution proceeds as follows. We start by reducing the problem of deciding whether C is a SGEC to the problem of deciding whether there is a winning strategy in a parity-Büchi game, using techniques from [7]. We then show how the latter can be solved by a reduction to positive Mean-Payoff parity games.

A parity-Büchi game is a two player game $G = \langle S_1, S_2, s_0, A_1, A_2, \delta_1, \delta_2, (\alpha, \beta)\rangle$ that is similar to a parity game, with the exception that the winning condition is composed of two conditions: α is a parity ranking function, and $\beta \subseteq Q$ is a set of *accepting states*. A play of G is winning for Player 1 iff it satisfies the parity condition α, and visits β infinitely often.

We start by describing a reduction from the problem of deciding whether C is a SGEC to the problem of solving a parity-Büchi game. First, we check that C is a GEC. If $\max_{s \in C} \{\alpha(s)\}$ is odd, then C is not a SGEC and we are done.

Consider the parity game $\mathcal{M}^{\mathsf{P}}|_C$. We obtain from $\mathcal{M}^{\mathsf{P}}|_C$ a parity-Büchi game G as follows. First, we change every state in $C^{\max}_{\text{even}}$ to a Büchi accepting sink (while keeping the parity rank).

For every state s of Player 2 that is not in $C^{\max}_{\text{even}}$, we replace s with the gadget in Figure 2.

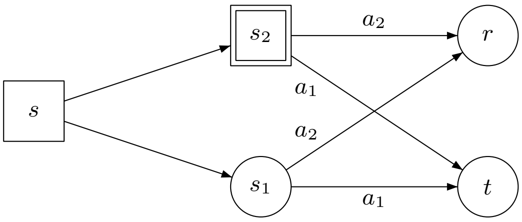

Fig. 2. Gadget for the reduction in Lemma 3. In $\mathcal{M}$, we have $\delta(s, a_1) = t$ and $\delta(s, a_2) = r$.

Formally, we add the states s_1, s_2, where s_1 is a Player 1 state and s_2 is a Player 2 state, whose successors are those of s (with the same available actions), and the successors of s are s_1 and s_2.

We set the parity ranks of the gadget to be $\alpha(s_1) = \alpha(s_2) = 0$, and for the Büchi objective, we set $s_2 \in \beta$ and $s_1 \notin \beta$.

We claim that C is a SGEC iff Player 1 wins in G from every state. For the first direction, assume C is a SGEC, and let f be a witness strategy. Thus, f is finite memory strategy that reaches $C^{\max}_{\text{even}}$ w.p. 1, and wins in the parity condition in every play that does not reach $C^{\max}_{\text{even}}$.

We obtain from f a strategy g for Player 1 in G as follows. g plays similarly to f, unless a state s_1 as in the gadget is reached, for some environment state s. Then g chooses the neighbor that minimizes the distance to $\mathrm{Attr}_{\mathrm{sys}}(C^{\max}_{\text{even}})$ (we assume w.l.o.g that in $\mathrm{Attr}_{\mathrm{sys}}(C^{\max}_{\text{even}})$, the strategy f leads surely to $C^{\max}_{\text{even}}$). We claim that g wins parity+Büchi in G. Indeed, consider a strategy g' for Player 2 in G, and consider the play ρ induced by g and g'. Note that g' induces a strategy in $\mathcal{M}^{\mathsf{P}}|_C$ by assigning each state $s \in S_2$ the action $g(s_2)$. Assume by way of contradiction that ρ is not winning for Player 1. Thus, either the Büchi condition or the parity conditions do not hold. If the Büchi condition does not hold, then after a finite prefix, for every environment state s, g' moves the play to s_1 in the gadget (since $s_2 \in \beta$). Thus, however, eventually Player 1 forces the play to $C^{\max}_{\text{even}}$, which are Büchi-winning sinks, and this the Büchi condition and the parity conditions are satisfied. Thus, the Büchi condition holds. If the parity condition does not hold, then the play does not reach $C^{\max}_{\text{even}}$. Since f is a witness strategy for C being a SGEC, then every play in $\mathcal{M}^{\mathsf{P}}|_C$ induced by f and does not reach $C^{\max}_{\text{even}}$ is parity winning. Thus, the play in G induces similar parity ranks, with the exception of padding 0

ranks within the gadgets. In particular, this play is also parity winning in G. Since this is true for every strategy g', we conclude that f is parity-Büchi winning in G.

conversely, assume that f is a parity-Büchi winning strategy in G. In addition, we assume that f is finite memory. Since parity-Büchi is an ω-regular winning condition, then Player 1 has a finite-memory winning strategy. The strategy f induces a strategy for Player 1 in $\mathcal{M}|_C$. We claim that this is a witness for C being a SGEC. Indeed, similarly to the above, it is easy to see that f is parity winning if $C^{\max}_{\text{even}}$ is not reached. It remains to prove that $C^{\max}_{\text{even}}$ is reached w.p. 1.

Since f has finite memory, then there exists $n \in \mathbb{N}$ such that for every state s in G, if Player 2 chooses t_1 from every environment state t for n steps, then f reaches $C^{\max}_{\text{even}}$. However, w.p. 1, a stochastic environment chooses the same n choices that f would have chosen in the above t_1 states. Thus, w.p. 1, f reaches $C^{\max}_{\text{even}}$.

This completes the reduction to parity-Büchi games.

Next, we reduce parity-Büchi games to Mean-payoff parity games by assigning every state in β payoff 1, and the rest payoff 0. Then, the goal is to win parity while having strictly positive long-run mean-payoff. These games can be solved in NP∩co-NP [7].

In addition, in case C is not a SGEC, our solution finds the winning states for Player 1, which are the super-good states.

B.3 Proof of Theorem 5

Intuitively, our algorithm works in two phases. First, for every odd rank k, we find the maximal SGEC whose maximal odd rank is k. Then, we choose among the SGEC the maximal ones. We start by describing a subroutine for the first phase.

Let d be the maximal parity rank in $\mathcal{M}$, and consider an odd rank $k \in \{-1, \dots, d\}$. We compute the maximal SGEC with maximal odd rank k as follows.

1. Compute the maximal EC decomposition $\mathcal{C}$.
2. For every EC $C \in \mathcal{C}$,
   (a) Let $odd_{>k}(C) = \{s \in C : \alpha(s) > k$ and $\alpha(s)$ is odd$\}$. If $odd_{>k}(C) \neq \emptyset$, remove $\mathrm{Attr}_{\mathrm{env}}(odd_{>k})$ from C, and go to (1).
   (b) Decide if C is a SGEC. If it is, return it. Otherwise, find the set W of super-good states, remove $\mathrm{Attr}_{\mathrm{env}}(C \setminus W)$ from C, and go to (1).

Next, we run this subroutine for every odd $k \in \{-1, \dots, d\}]$ to obtain SGEC $C_1, ..., C_m$. Finally, for every C_i, C_j, if $C_i \subseteq C_j$, we remove C_j from the list.

Clearly this algorithm has polynomially many iterations, and in each iteration we solve an NP∩co-NP problem, as per Lemma 3. Thus, the algorithm solves the problem in NP∩co-NP.

It remains to prove the correctness of the algorithm. By Lemma 3, every component that is returned in the subroutine is a SGEC. Consider a SGEC C with maximal odd rank k. In iteration k of the algorithm, none of the states of C are removed in steps 2a and 2b. Thus, the subroutine returns a SGEC D such that $C \subseteq D$. Finally, by Lemma 2, if $C_i \cap C_j \neq \emptyset$, then there exists a SGEC E such that $C_i \cup C_j \subseteq E$. Thus, E is also returned in the list, and will replace C_i and C_j. We conclude that the returned list contains exactly the maximal SGEC of $\mathcal{M}$.

B.4 Proof of Lemma 5

Let f be a finite-memory strategy with memory M. Consider a history $h \in S^* \times S_1$. Let $m \in M$ be the memory element that f reaches after reading h, we define the strategy f_h to be f when starting from m. Note that the set $F = \{f_h : h \in S^* \times S_1\}$ is finite, since M is finite. We claim that there exists $g \in F$ that satisfies the conditions of the lemma.

Indeed, assume by way of contradiction that for every $g \in F$ we have that $\Pr_{\mathcal{M}^s}(\text{inf}(g) = C) < 1$. Thus, there exists $\epsilon > 0$ such that $\Pr_{\mathcal{M}^s}(\text{inf}(g) = C) < 1 - \epsilon$ for every $g \in F$. It follows that there exists $\delta > 0$ such that for every history h, w.p. at least δ the strategy f_h from s reaches either a state $t \notin C$ or a state $t' \in C$ such that there exists $s' \in C$ that is not reachable from t' under f_h. Since δ is independent of h, and since this is true for every h, we get that $\Pr_{\mathcal{M}}(\text{inf}(f) = C) = 0$, in contradiction to the assumption.

Let $G = \{g \in F : \Pr_{\mathcal{M}^s}(\text{inf}(g) = C) = 1\}$, then we conclude that $G \neq \emptyset$. Assume by way of contradiction that for every $g \in G$ it holds that there exists a play of g from some state $s \in C$ that leaves C (which happens after a finite number of steps). Thus, there exists some $\delta > 0$ such that w.p. at least δ (independent of g), for every $g' \in G$ and every state $s \in C$ a play of g' leaves C (since every $g' \in G$ visits every state of C w.p. 1). This contradicts the fact that $\Pr_{\mathcal{M}^s}(\text{inf}(g) = C) = 1$. We conclude that there exists $g \in G$ such that every play of g stays in C forever.

In addition, since f is parity winning, and the parity condition is independent of the history, then g is parity winning too.

B.5 Proof of Lemma 6

Let g be a strategy obtained as per Lemma 5. Thus, $\Pr_{\mathcal{M}^s}(\text{inf}(g) = C) = 1$ for every $s \in C$, every play of g from s stays in C, and g is parity winning. We show that C is a SGEC by showing that g is a witness thereof. Indeed, w.p. 1 g visits every state of C, and in particular g reaches $C^{\max}_{\text{even}}$ w.p. 1. In addition, g is parity-winning, so every play of g is parity winning, in particular plays that do not reach $C^{\max}_{\text{even}}$. □

C Applications

C.1 Automata, and the Boolean Synthesis Problem

An *automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, and α is an acceptance condition. We define some acceptance conditions below. The automaton $\mathcal{A}$ may run on finite or infinite words. A run of $\mathcal{A}$ on a finite word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n \in \Sigma^*$ is a sequence of states $r = r_0, r_1, \ldots, r_n$ such that $r_{i+1} \in \delta(r_i, \sigma_{i+1})$ for all $0 \leq i < n$. When w is infinite, so is a run of $\mathcal{A}$ on it. For an infinite run r, we denote by $\text{inf}(r)$ the set of states that r visits infinitely often.

We consider two acceptance conditions. When $\mathcal{A}$ runs on finite words, we have that $\alpha \subseteq Q$ is a set of accepting states. Then, a finite run $r_0, r_1, \ldots, r_n$ is accepting if $r_n \in \alpha$. When $\mathcal{A}$ runs on infinite words, then $\alpha : Q \to \{0, ..., d\}$ is a *parity* acceptance condition. For a state $q \in Q$, we refer to $\alpha(q)$ as the *rank* of q. Then, an infinite run r is accepting if $\max\{\alpha(q) : q \in \text{inf}(r)\}$ is even.

The automata we consider are universal. Thus, a word $w \in \Sigma^\omega$ is accepted if all the runs of $\mathcal{A}$ on it are accepting. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. If $|\delta(q,\sigma)| = 1$ for every $q \in Q$ and $\sigma \in \Sigma$, we say that $\mathcal{A}$ is *deterministic*. Note that in this case, $\mathcal{A}$ has exactly one run on every word.

The classical solution to the Boolean synthesis problem proceeds as follows. Consider a specification DPW $\mathcal{A} = \langle 2^I \times 2^O, Q, q_0, \delta, \alpha \rangle$. We obtain from $\mathcal{A}$ a parity game $\mathcal{G} = \langle Q \times 2^I, Q, q_0, 2^O, 2^I, \ \delta_1, \delta_2, \alpha' \rangle$, where $\delta_1(\langle q,i \rangle, o) = \delta(q, i \cup o)$, and $\delta_2(q,i) = \langle q,i \rangle$. Thus, Player 2, the environment, controls the inputs and his actions correspond to assignments to the input signals. His states are the states of $\mathcal{A}$, and he moves to states that maintain the assignment he gives to the input signals. Then, Player 1, the system, controls the outputs and his actions correspond to assignments to the output signals. He moves in states that maintain the assignment to the input signals given by Player 2, and his transitions update the state of $\mathcal{A}$. Then, α' in induced by α. Formally, for every $q \in Q$ and for every $i \in 2^I$, we have that $\alpha'(q) = \alpha'(\langle q,i \rangle) = \alpha(q)$. It is not hard to see that a winning strategy for Player 1 in $\mathcal{G}$ induces a transducer that realizes $\mathcal{A}$ [18]. Finding a winning strategy for Player 1 amounts to solving a turn-based parity game, whose complexity is NP∩co-NP. Alternatively, deterministic algorithms for solving parity games run in time polynomial in the number of states, and singly-exponential in the number of parity ranks. When the starting point is an LTL formula ψ, the translation to a DPW involves a doubly-exponential blow up, but the index of the DPW is only exponential, so the problem is 2EXPTIME-complete [20].

C.2 Synthesis with Penalties

Let $\mathcal{U}_i = \langle 2^I \times 2^O, Q^i, q_0^i, \delta^i, \alpha^i \rangle$ Let $S = Q \times S_1 \times \cdots S_m$ and $s_0 = \langle q_0, q_0^1, \ldots, q_0^m \rangle$. We define the parity-MDP $\mathcal{M} = \langle S \times 2^I, S, s_0, 2^O, 2^I, \delta_1, \delta_2, \mathrm{P}, cost, \alpha' \rangle$ where for every $s = \langle q, q^1, ..., q^m \rangle \in S$, $i \in 2^I$, and $o \in 2^O$, we have the following. The transition functions are $\delta_1(\langle s,i \rangle, o) = \langle \delta(q, i \cup o), \delta^1(q^1, i \cup o), \ldots, \delta^m(q^m, i \cup o) \rangle$, and $\delta_2(s,i) = \langle s,i \rangle$, the cost function is given by $cost(\langle s,i \rangle) = 0$ and $cost(s) = \sum_{j: q^j \in \alpha^j} \gamma(j)$, for the penally function γ, and the acceptance condition is $\alpha(s) = \alpha(\langle s,i \rangle) = \alpha(q)$. Finally, we assume that the environment behaves uniformly. That is, in every step it outputs every $i \subseteq I$ with probability $2^{-|I|}$. Thus, $\mathrm{P}(s,i) = 2^{-|I|}$. This assumption can easily be replaced by a different probabilistic model.

It is easy to see that a winning strategy for Player 1 in $\mathcal{M}$ corresponds to a transducer that realizes $\mathcal{A}$, and that the cost of every computation is the average penalty along the computation. Thus, a solution to the synthesis with penalties problem amounts to solving $\mathcal{M}$. The size of $\mathcal{M}$ is polynomial in the size of the automata $\mathcal{A}, \mathcal{U}_1, \ldots \mathcal{U}_m$, and is exponential in m. However, we observe that the role of $\mathcal{U}_1, \ldots, \mathcal{U}_m$ is only for the purpose of costs, and does not affect the parity constraints. Thus, we can solve the problem in NP∩co-NP in the size of the automata, and in time singly-exponential in m. Finally, if $\mathcal{A}$ is obtained by translating an LTL formula ψ into a DPW, then similarly to the case of Boolean synthesis, we can solve the problem in times doubly-exponential in the length of ψ, polynomial in $\mathcal{U}_1, \ldots, \mathcal{U}_m$, and singly-exponential in m.

C.3 Proof of Theorem 7

We identify a subset $i \subseteq I$ with its characteristic function $i : I \to \{0,1\}$.

Consider the DPW $\mathcal{A} = \langle 2^I \times 2^O, Q, q_0, \delta, \alpha\rangle$. We obtain from $\mathcal{A}$ the UPW $\mathcal{A}' = \langle 2^I \times 2^I \times 2^O, Q, q_0, \delta', \alpha\rangle$ with δ' defined as follows. Consider a letter $\langle i, x, o\rangle \in 2^I \times 2^I \times 2^O$. We think of i and o as truth assignments for the input and output signals, respectively, and we think of x as a set of sensed signals. Consider the set ${}^{i}\!/\!_{x} = \{j \in 2^I : \forall p \in x,\ j(p) = i(p)\}$. Intuitively, ${}^{i}\!/\!_{x}$ is the set of input assignments that agree with i on all the signals in x. For a state $q \in Q$, we define $\delta'(q, \langle i, x, o\rangle) = \{\delta(q, (j, o)) : j \in {}^{i}\!/\!_{x}\}$.

Intuitively, when thinking of $\mathcal{A}'$ as a game between the system and the environment, then at each step, the system chooses a set of sensed inputs x and an output o. Then, the environment chooses a set of inputs i, but in the next step the system can only see the inputs in i that are sensed in x, and thus moves universally with every input that agrees with i on the sensed inputs in x.

We proceed to determinize $\mathcal{A}'$ to a DPW $\mathcal{D} = \langle 2^I \times 2^I \times 2^O, S, \rho, s_0, \beta\rangle$. We then obtain from $\mathcal{D}$ a parity game, as described above, with Player 1 (the system) controlling the set of sensed inputs and the output, and Player 2 (the environment) controlling the concrete inputs. Formally, the game $G_{\mathcal{D}} = \langle S_1 \cup S_2, \textsc{start}, A_1, A_2, \delta_1, \delta_2, \beta'\rangle$ is defined as follows. The states are $S_1 = (S \times 2^I \times 2^I) \cup \{\textsc{start}\}$ and $S_2 = S \times 2^I$. The actions for Player 1 in every state are $A_1 = 2^I \times 2^O$ and are $A_2 = 2^I$ for Player 2 (we omit the state as the available actions are independent of the state). The transition function is defined as follows. For a state $\langle s, x, i\rangle \in S_1$ and action $\langle x', o\rangle \in A_1$ we have $\delta_1(\langle s, x, i\rangle, \langle x', o\rangle) = \langle \rho(s, \langle i, x, o\rangle), x'\rangle$ as well as $\delta_1(\textsc{start}, \langle x', o\rangle) = \langle s_0, x'\rangle$. For a state $\langle s, x\rangle \in S_2$ and action $i \in A_2$ we have $\delta_2(\langle s, x\rangle, i) = \langle s, x, i\rangle$.

Intuitively, the state $\langle s, x, i\rangle \in S_1$ represents that $\mathcal{D}$ is in state s, the system has chosen to sense the signals in x, and the environment gave the concrete input i. Then, the action $\langle x', o\rangle$ means that the system responded with output o, and chose to sense x' in the next step, taking the game to the state $\langle s', x'\rangle$, where $s' = \rho(s, \langle i, x, o\rangle)$. Then, in state $\langle s', x'\rangle$, the environment chooses a new concrete input i'.

We define the acceptance condition β' as follows. For every $s \in S$ and $i, x \in 2^I$, we have $\beta'(\langle s, x, i\rangle) = \beta'(\langle s, x\rangle) = \beta(s)$, and we arbitrarily set $\beta'(\textsc{start}) = 0$ (since $\textsc{start}$ is visited only once, this has no effect).

Note that crucially, for every $j, j' \in {}^{i}\!/\!_{x}$, the behavior of $G_{\mathcal{D}}$ from state $\langle s, x, j\rangle$ is identical to the behavior from $\langle s, x, j'\rangle$. This follows from the universal transitions in $\mathcal{A}'$. Thus, once Player 1 chooses x, the inputs that are not sensed do not play a role. This captures the fact that every winning strategy for the system must only rely on the values j assigns to the sensed inputs x.

Finally, the parity-MDP $\mathcal{M}$ is obtained from $G_{\mathcal{D}}$ by fixing Player 2 with a uniform-stochastic strategy and adding costs according to the number of sensed inputs at each state. Recall that the actions of Player 2 are 2^I. Thus, in state $\langle s, x\rangle \in S_2$, the probability of Player 2 playing $j \in 2^I$ is $2^{-|I|}$. Note that by our observation above, every $j, j' \in {}^{i}\!/\!_{x}$, induce the same transitions. Thus, the probability of transition from state $\langle s, x\rangle$ to $\langle s, x, j\rangle$ is $2^{-|x|}$.

The cost function assigns cost $|x|$ to states $\langle s, x\rangle$ and $\langle s, x, j\rangle$, for every $s \in S$ and $j \in 2^I$.

We now proceed to analyze the correctness of the construction. Consider a (not necessarily finite) transducer $\mathcal{T} = \langle I, O, T, t_0, \tau, \rho \rangle$ that realizes the specification $\mathcal{A}$. We identify with $\mathcal{T}$ a strategy $f_{\mathcal{T}}$ for $\mathcal{M}$ as follows. In state START we have $f_{\mathcal{T}}(\text{START}) = \langle sensed(t_0), \rho(t_0) \rangle$. Then, the strategy $f_{\mathcal{T}}$ keeps track of the state of $\mathcal{T}$ as follows. When $\mathcal{T}$ is in state t, and the state of the game is $\langle s, x, i \rangle$, let $t' = \tau(t, i)$. Then, we have that $f_{\mathcal{T}}(\langle s, x, i \rangle) = \langle sensed(t'), \rho(t') \rangle$. Observe that $f_{\mathcal{T}}$ is essentially implemented by the transducer $\mathcal{T}$. In particular, if $\mathcal{T}$ has finite state space, then f has finite memory.

We claim that $cost_{\text{sure}}(f_{\mathcal{T}}) = scost(\mathcal{T})$. We start by showing that $f_{\mathcal{T}}$ is sure winning in $\mathcal{M}$ (equivalently, that it is a winning strategy for Player 1 in $G_{\mathcal{D}}$). Consider an input sequence $\pi \in (2^I)^*$, let q and t be the states that $\mathcal{A}$ and $\mathcal{T}$ reach, respectively, when they interact on π. let $x = sensed(t)$, then for every $i, j \in 2^I$ such that $j \in {}^{i}\!/_{x}$ we have that $\tau(t, i) = \tau(t, j)$. Thus, the behavior of $\mathcal{T}$ from $\delta(q, i \cup \rho(t))$ and from $\delta(q, j \cup \rho(t))$ is the same. It follows that $\mathcal{T}$ induces a realizing strategy for the UPW $\mathcal{A}'$ (and hence a winning strategy for $G_{\mathcal{D}}$), where the additional 2^I component in the alphabet represents the sensing of the current state of $\mathcal{T}$. However, this is exactly the behavior prescribed by $f_{\mathcal{T}}$, so $f_{\mathcal{T}}$ is winning in $G_{\mathcal{D}}$.

Next, observe that by the above, for every input sequence $\pi \in (2^I)^\omega$, the (prefix of the) play of $G_{\mathcal{D}}$ induced by Player 1 playing $f_{\mathcal{T}}$ and Player 2 playing π is $r = \text{START}, \langle s_1, x_1 \rangle, \langle s_1, x_1, \pi_1 \rangle, ..., \langle s_m, x_m \rangle, \langle s_m, x_m, \pi_m \rangle$, and we have that $cost_m(f_{\mathcal{T}}, \pi) = \frac{1}{2m+1}(\sum_{k=1}^{m} 2 \cdot |x_k|)$, while for the run $r = t_1, t_2, ..., t_m$ of $\mathcal{T}$ on the first m letters of π we have that $scost(r) = \frac{1}{m}\sum_{k=1}^{m} scost(t_k)$. By the definition of $\mathcal{T}$ and $\mathcal{M}$, we have $scost(t_k) = |x_k| = cost(\langle s_k, x_k \rangle) = cost(\langle s_k, x_k, \pi_k \rangle)$. Moreover, the probabilities of $\mathcal{M}$ imply that every π such that $|\pi| = m$ is played w.p. $|2^I|^{-m}$. Thus, by taking $m \to \infty$, we get $cost_{\text{sure}}(f_{\mathcal{T}}) = scost(\mathcal{T})$.

Since this is true for every realizing transducer $\mathcal{T}$, it follows that $cost_{\text{sure}}(\mathcal{M}) \leq scost_{I/O}(L(\mathcal{A}))$.

Conversely, consider a strategy f for $\mathcal{M}$. A-priori, f can behave differently in states $\langle s, x, i \rangle$ and $\langle s, x, j \rangle$ for $j \in {}^{i}\!/_{x}$. However, as we observed above, the construction of $\mathcal{A}'$ (and thus of $\mathcal{D}$) implies that f cannot decrease its cost by doing so, since the behavior of $\mathcal{A}'$ is the same in both states. Thus, we can assume w.l.o.g that f only depends on the values i assigns to the sensed inputs x. Now, f induces a (possibly infinite) transducer $\mathcal{T}_f$ in an obvious manner - whenever f outputs $\langle x, o \rangle$, the transducer outputs o. Similar arguments as the converse direction show that $cost_{\text{sure}}(f) = scost(\mathcal{T}_f)$, and thus $cost_{\text{sure}}(\mathcal{M}) \geq scost_{I/O}(L(\mathcal{A}))$, and we are done.