Synthesis from LTL Specifications with
Mean-Payoff Objectives

Aaron Bohy[1], Véronique Bruyère[1], Emmanuel Filiot[2], and Jean-François Raskin[3]

[1]Université de Mons   [2]Université Paris-Est Créteil   [3]Université Libre de Bruxelles

Abstract. The classical LTL synthesis problem is purely qualitative: the given LTL specification is realized or not by a reactive system. LTL is not expressive enough to formalize the correctness of reactive systems with respect to some quantitative aspects. This paper extends the *qualitative* LTL synthesis setting to a *quantitative* setting. The alphabet of actions is extended with a weight function ranging over the rational numbers. The value of an infinite word is the mean-payoff of the weights of its letters. The synthesis problem then amounts to automatically construct (if possible) a reactive system whose executions all satisfy a given LTL formula and have mean-payoff values greater than or equal to some given threshold. The latter problem is called LTL_{MP} synthesis and the LTL_{MP} realizability problem asks to check whether such a system exists. We first show that LTL_{MP} realizability is not more difficult than LTL realizability: it is 2ExpTime-Complete. This is done by reduction to two-player mean-payoff parity games. While infinite memory strategies are required to realize LTL_{MP} specifications in general, we show that ϵ-optimality can be obtained with finite memory strategies, for any $\epsilon > 0$. To obtain an efficient algorithm in practice, we define a Safraless procedure to decide whether there exists a finite-memory strategy that realizes a given specification for some given threshold. This procedure is based on a reduction to two-player energy safety games which are in turn reduced to safety games. Finally, we show that those safety games can be solved efficiently by exploiting the structure of their state spaces and by using antichains as a symbolic data-structure. All our results extend to multi-dimensional weights. We have implemented an antichain-based procedure and we report on some promising experimental results.

1 Introduction

Formal specifications of reactive systems are usually expressed using formalisms like the linear temporal logic (LTL), the branching time temporal logic (CTL), or automata formalisms like Büchi automata. Those formalisms allow the specifier to express *Boolean properties* (called *qualitative properties* in the sequel) in the sense that a reactive system either conforms to them, or violates them. Additionally to those qualitative formalisms, there is a clear need for another family of formalisms that are able to express *quantitative properties* of reactive systems. Abstractly, a quantitative property can be seen as a function that maps an execution of a reactive system to a numerical value. For example, in a client-server application, this numerical value could be the mean number of steps that separate the time at which a request has been emitted by a client and the time at which this request has been granted by the server along an execution. Quantitative properties are concerned with a large variety of aspects like quality of service, bandwidth, energy consumption,... But quantities are also useful to compare the merits of alternative solutions, e.g. we may prefer a solution in which the quality of service is high and the energy consumption is low. Currently, there is a large effort of the research community with the objective to lift the theory of formal verification and synthesis from the *qualitative world* to the richer *quantitative world* [20] (see related works for more details). In this paper, we consider mean-payoff and energy objectives. The alphabet of actions is extended with a weight function ranging over the rational numbers. A mean-payoff objective is a set of infinite words such that the mean value of the weights of their letters is greater than or equal to a given rational threshold [26], while an energy objective is parameterized by a non-negative initial energy level c_0 and contains all the words whose finite prefixes have a sum of weights greater than or equal to $-c_0$ [7].

In this paper, we participate to this research effort by providing theoretical complexity results, practical algorithmic solutions, and a tool for the automatic synthesis of reactive systems from *quantitative specifications* expressed in the linear time temporal logic LTL extended with (multi-dimensional) mean-payoff and (multi-dimensional) energy objectives. To illustrate our contributions, let us consider the following specification of a controller that should grant exclusive access to a resource to two clients.

Example 1. A client requests access to the resource by setting to true its request signal (r_1 for client 1 and r_2 for client 2), and the server grants those requests by setting to true the respective grant signal g_1 or g_2. We want to synthetize a server that eventually grants any client request, and that only grants one request at a time. This can be formalized in LTL where the signals in $I = \{r_1, r_2\}$ are controlled by the environment (the two clients), and the signals in $O = \{g_1, w_1, g_2, w_2\}$ are controlled by the server:

$$\phi_1 = \Box(r_1 \rightarrow X(w_1 U g_1))$$
$$\phi_2 = \Box(r_2 \rightarrow X(w_2 U g_2))$$
$$\phi_3 = \Box(\neg g_1 \vee \neg g_2)$$
$$\phi = \phi_1 \wedge \phi_2 \wedge \phi_3$$

Intuitively, ϕ_1 (resp. ϕ_2) specifies that any request of client 1 (resp. client 2) must be eventually granted, and in-between the waiting signal w_1 (resp. w_2) must be high. Formula ϕ_3 stands for mutual exclusion.

The formula ϕ is realizable. One possible strategy for the server is to alternatively assert w_2, g_1 and w_1, g_2, i.e. alternatively grant client 1 and client 2. While this strategy is formally correct, as it realizes the formula ϕ against all possible behaviors of the clients, it may not be the one that we expect. Indeed, we may prefer a solution that does not make unsollicited grants for example. Or, we may prefer a solution that gives, in case of request by both clients, some priority to client 2's request. In the later case, one elegant solution would be to associate a cost equal to 2 when w_2 is true and a cost equal to 1 when w_1 is true. This clearly will favor solutions that give priority to requests from client 2 over requests from client 1. We will develop several other examples in the paper and describe the solutions that we obtain automatically with our algorithms.

Contributions We now detail our contributions and give some hints about the proofs. In Section 2, we define the realizability problems for LTL$_{\mathsf{MP}}$ (LTL extended with mean-payoff objectives) and LTL$_{\mathsf{E}}$ (LTL extended with energy objectives), and give some examples. In Section 3, we show that, as for the LTL realizability problem, both the LTL$_{\mathsf{MP}}$ and LTL$_{\mathsf{E}}$ realizability problems are 2ExpTime-Complete. As the proof of those three results follow a similar structure, let us briefly recall how the 2ExpTime upper bound of the classical LTL realizability problem is established in [23]. The formula is first turned into an equivalent nondeterministic Büchi automaton, which is then transformed into a deterministic parity automaton using Safra's construction. The latter automaton can be seen as a two-player parity game in which Player 1 wins if and only if the formula is realizable. For the LTL$_{\mathsf{MP}}$ realizability problem, our construction follows the same structure, except that we go to a two-player parity game with an additional mean-payoff objective, and for the LTL$_{\mathsf{E}}$ realizability problem, we need to consider a parity game with an additional energy objective. By a careful analysis of the complexity of all the steps involved in those two constructions, we build, on the basis of results in [12] and [14], solutions that provide the announced 2ExpTime upper bound.

It is known that winning mean-payoff parity games may require infinite memory strategies, but there exist ϵ-optimal finite-memory strategies [14]. In contrast, for energy parity games, it is known that finite-memory optimal strategies exist [12]. In Section 3, we show that those results transfer to LTL$_{\mathsf{MP}}$ (resp. LTL$_{\mathsf{E}}$) realizability problems thanks to the reduction of these problems to mean-payoff (resp. energy) parity games. Furthermore, we show that under finite-memory strategies, LTL$_{\mathsf{MP}}$ realizability is in fact equivalent to LTL$_{\mathsf{E}}$ realizability: a specification is MP-realizable under finite-memory strategies if and only if it is E-realizable, by simply shifting the weights of the signals by the threshold value. Because finite-memory strategies are more interesting in practice, we thus concentrate on the LTL$_{\mathsf{E}}$ realizability problem in rest of the paper.

Safra's construction has so far resisted all the practical attempts to implement it efficiently, see for example [1]. We develop in Section 4, following [22], a Safraless procedure for the LTL$_{\mathsf{E}}$ realizability problem, that is based on a reduction to a safety game, with the nice property to transform a quantitative objective into a simple qualitative objective. The main building blocks of this procedure are as follows. *(1)* Instead of transforming an LTL formula into a deterministic parity automaton, we prefer to use a universal co-Büchi automaton as proposed in [22]. To deal with the energy objectives, we thus transform the formula into a universal co-Büchi energy automaton for some initial credit c_0, which requires that all runs on an input word w visit finitely many accepting states and the energy level of w is always positive starting from the initial energy level c_0. *(2)* By strenghtening the co-Büchi condition into a K-co-Büchi condition (as

done in [24,18]), where at most K accepting states can be visited by each run, we then go to an energy safety game. We show that for sufficiently large value K and initial credit c_0, this reduction is complete.*(3)* Any energy safety game is equivalent to a safety game, as shown in [9].

Finally, in Section 5, we discuss some implementation issues. The proposed Safraless construction has two main advantages. Firstly, the search for winning strategies for $\mathsf{LTL_E}$ realizability can be incremental on K and c_0 (avoiding in practice to consider the large theoretical bounds $\mathbb{K}$ and $\mathbb{C}$ that ensure completeness). Secondly, the state space of the safety game can be partially ordered and solved by a backward fixpoint algorithm. Since the latter manipulates sets of states closed for this order, it can be made efficient and symbolic by working only on the antichain of their maximal elements. As described in Section 6, our results can be extended to the multi-dimensional case, i.e. tuples of weights. All the algorithms have been implemented in our tool Acacia+ [4], and promising experimental results are reported in Section 7.

Related works The LTL synthesis problem has been first solved in [23], Safraless approaches have been proposed in [21,22,24,18], and implemented in prototypes of tools [21,17,16,4]. All those works only treat plain qualitative LTL, and not the quantitative extensions considered in this article.

Mean-payoff games [26] and energy games [7,9], extensions with parity conditions [14,12,8], or multi-dimensions [13,15] have recently received a large attention from the research community. The use of such game formalisms has been advocated in [3] for specifying quantitative properties of reactive systems. Several among the motivations developed in [3] are similar that our motivations for considering quantitative extensions of LTL. All these related works make the assumption that the game graph is given explicitly (and not implicitly using an LTL formula), as in our case.

In [5], Boker et al. introduce extensions of linear and branching time temporal logics with operators to express constraints on values accumulated along the paths of a weighted Kripke structure. One of their extensions is similar to $\mathsf{LTL_{MP}}$. However the authors of [5] only study the complexity of model-checking problems whereas we consider realizability and synthesis problems.

2 Problem statement

2.1 Preliminaries

Linear temporal logic – The formulas of linear temporal logic (LTL) are defined over a finite set P of atomic propositions. The syntax is given by the grammar:

$$\phi \ ::= \ p \mid \phi \vee \phi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi\mathsf{U}\phi \qquad p \in P$$

The notations true, false, $\phi_1 \wedge \phi_2$, $\Diamond\phi$ and $\Box\phi$ are defined as usual. LTL formulas ϕ are interpreted on infinite words $u = \sigma_0\sigma_1\sigma_2\cdots \in (2^P)^\omega$ via a satisfaction relation $u \models \phi$ inductively defined as:

$\quad u \models p \qquad$ if $p \in \sigma_0$
$\quad u \models \phi_1 \vee \phi_2$ if $u \models \phi_1$ or $u \models \phi_2$
$\quad u \models \neg\phi \qquad$ if $u \not\models \phi$
$\quad u \models \mathsf{X}\phi \qquad$ if $\sigma_1\sigma_2\ldots \models \phi$
$\quad u \models \phi_1\mathsf{U}\phi_2$ if $\exists n \geq 0 \cdot \sigma_n\sigma_{n+1}\ldots \models \phi_2$ and $\forall i, 0 \leq i < n \cdot \sigma_i\sigma_{i+1}\ldots \models \phi_1$.

Given an LTL formula ϕ, we denote by $[\![\phi]\!]$ the set of words u such that $u \models \phi$.

LTL *Realizability and synthesis –* The realizability problem for LTL is best seen as a game between two players. Let ϕ be an LTL formula over the set $P = I \uplus O$ partitioned into I the set of *input signals* controlled by Player I (the environment), and O the set of *output signals* controlled by Player O (the controller). With this partition of P, we associate the three following alphabets: $\Sigma_P = 2^P$, $\Sigma_O = 2^O$, and $\Sigma_I = 2^I$.

The realizability game is played in turns. Player O starts by giving $o_0 \in \Sigma_O$, Player I responds by giving $i_0 \in \Sigma_I$, then Player O gives $o_1 \in \Sigma_O$ and Player I responds by $i_1 \in \Sigma_I$, and so on. This game lasts forever and the outcome of the game is the infinite word $(o_0 \cup i_0)(o_1 \cup i_1)(o_2 \cup i_2)\cdots \in \Sigma_P^\omega$.

The players play according to *strategies*. A strategy for Player O is a mapping $\lambda_O : (\Sigma_O\Sigma_I)^* \to \Sigma_O$, while a strategy for Player I is a mapping $\lambda_I : (\Sigma_O\Sigma_I)^*\Sigma_O \to \Sigma_I$. The outcome of the strategies λ_O and λ_I is the word $\mathsf{Outcome}(\lambda_O, \lambda_I) = (o_0 \cup i_0)(o_1 \cup i_1)\ldots$ such that $o_0 = \lambda_O(\epsilon)$, $i_0 = \lambda_I(o_0)$ and for all

$k \geq 1$, $o_k = \lambda_O(o_0 i_0 \ldots o_{k-1} i_{k-1})$ and $i_k = \lambda_I(o_0 i_0 \ldots o_{k-1} i_{k-1} o_k)$. We denote by $\mathsf{Outcome}(\lambda_O)$ the set of all outcomes $\mathsf{Outcome}(\lambda_O, \lambda_I)$ with λ_I any strategy of Player I. We let Π_O (resp. Π_I) be the set of strategies for Player O (resp. Player I).

Given an LTL formula ϕ (the specification), the LTL *realizability problem* is to decide whether there exists a strategy λ_O of Player O such that $\mathsf{Outcome}(\lambda_O, \lambda_I) \models \phi$ against all strategies λ_I of Player I. If such a *winning* strategy exists, we say that the specification ϕ is *realizable*. The LTL *synthesis problem* asks to produce a strategy λ_O that realizes ϕ, when it is realizable.

Moore machines – It is known that the LTL realizability problem is 2ExpTime-Complete and that finite-memory strategies suffice in case of realizability [23]. A strategy λ_O of Player O is *finite-memory* if there exists a right-congruence $\sim$ on $(\Sigma_O \Sigma_I)^*$ of finite index such that $\lambda_O(u) = \lambda_O(u')$ for all $u \sim u'$. It is equivalent to say that it can be described by a *Moore machine* $\mathcal{M} = (M, m_0, \alpha_U, \alpha_N)$ defined as follows. The non-empty set M is the finite memory[1] of $\mathcal{M}$ and m_0 is its initial memory state. The memory update function $\alpha_U : M \times \Sigma_I \to M$ modifies the current memory state at each $i \in \Sigma_I$ emitted by Player I, and the next-move function $\alpha_N : M \to \Sigma_O$ indicates which $o \in \Sigma_O$ is proposed by Player O given the current memory state. The function α_U is naturally extended to words $u \in \Sigma_I^*$. The *language* of $\mathcal{M}$, denoted by $L(\mathcal{M})$, is the set of words $u = (o_0 \cup i_0)(o_1 \cup i_1) \cdots \in \Sigma_P^\omega$ such that $o_0 = \alpha_N(m_0)$ and for all $k \geq 1$, $o_k = \alpha_N(\alpha_U(m_0, i_0 \ldots i_{k-1}))$. The size $|\mathcal{M}|$ of a Moore machine is defined as the size $|M|$ of its memory. Therefore, with these notations, an LTL formula is realizable iff there exists a Moore machine such that $L(\mathcal{M}) \subseteq \llbracket \phi \rrbracket$.

Theorem 1 ([23]). *The* LTL *realizability problem is 2ExpTime-Complete and any realizable* LTL *formula is realizable by a finite-memory strategy.*

2.2 Synthesis with mean-payoff objectives

LTL$_{\mathsf{MP}}$ *realizability and synthesis –* Consider a finite set P partitioned as $I \uplus O$. Let $\mathsf{Lit}(P)$ be the set $\{p \mid p \in P\} \cup \{\neg p \mid p \in P\}$ of literals over P, and let $w : \mathsf{Lit}(P) \to \mathbb{Z}$ be a *weight function* where positive numbers represent rewards[2]. This function is extended to Σ_I (resp. Σ_O) as follows: $w(i) = \Sigma_{p \in i} w(p) + \Sigma_{p \in I \setminus \{i\}} w(\neg p)$ for $i \in \Sigma_I$ (resp. $w(o) = \Sigma_{p \in o} w(p) + \Sigma_{p \in O \setminus \{o\}} w(\neg p)$ for $o \in \Sigma_O$). In this way, it can also be extended to Σ_P as $w(o \cup i) = w(o) + w(i)$ for all $o \in \Sigma_O$ and $i \in \Sigma_I$.[3] In the sequel, we denote by $\langle P, w \rangle$ the pair given by the finite set P and the weight function w over $\mathsf{Lit}(P)$; we also use the weighted alphabet $\langle \Sigma_P, w \rangle$.

Consider an LTL formula ϕ over $\langle P, w \rangle$ and an outcome $u = (o_0 \cup i_0)(o_1 \cup i_1) \cdots \in \Sigma_P^\omega$ produced by Players I and O. We associate a *value* $\mathsf{Val}(u)$ with u that captures the two objectives of Player O of both satisfying ϕ and achieving a mean-payoff objective. For each $n \geq 0$, let $u(n)$ be the prefix of u of length n. We define the *energy level* of $u(n)$ as $\mathsf{EL}(u(n)) = \sum_{k=0}^{n-1} w(o_k) + w(i_k)$. We then assign to u a *mean-payoff value* equal to $\mathsf{MP}(u) = \liminf_{n \to \infty} \frac{1}{n} \mathsf{EL}(u(n))$. Finally we define the value of u as:

$$\mathsf{Val}(u) = \begin{cases} \mathsf{MP}(u) \text{ if } u \models \phi \\ -\infty \text{ otherwise.} \end{cases}$$

Given an LTL formula ϕ over $\langle P, w \rangle$ and a threshold $\nu \in \mathbb{Q}$, the LTL$_{\mathsf{MP}}$ *realizability problem (resp.* LTL$_{\mathsf{MP}}$ *realizability problem under finite memory)* asks to decide whether there exists a strategy (resp. finite-memory strategy) λ_O of Player O such that $\mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I)) \geq \nu$ against all strategies λ_I of Player I, in which we say that ϕ is MP*-realizable (resp.* MP*-realizable under finite memory)* . The LTL$_{\mathsf{MP}}$ *synthesis problem* is to produce such a winning strategy λ_O for Player O. Therefore the aim is to achieve two objectives: *(i)* realizing ϕ, *(ii)* having a long-run average reward greater than the given threshold.

---

[1] The memory M is the set of equivalence classes for $\sim$.

[2] We use weights at several places of this paper. In some statements and proofs, we take the freedom to use *rational* weights as it is equivalent up to rescaling. However we always assume that weights are integers encoded in binary for complexity results.

[3] The decomposition of $w(o \cup i)$ as the sum $w(o) + w(i)$ emphasizes the partition of P as $I \uplus O$ and will be useful in some proofs.

Optimality – Given ϕ an LTL formula over $\langle P, w \rangle$, the *optimal value* (for Player O) is defined as

$$\nu_\phi = \sup_{\lambda_O \in \Pi_O} \inf_{\lambda_I \in \Pi_I} \mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I)).$$

For a real-valued $\epsilon \geq 0$, a strategy λ_O of Player O is ϵ-*optimal* if $\mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I)) \geq \nu_\phi - \epsilon$ against all strategies λ_I of Player I. It is *optimal* if it is ϵ-optimal with $\epsilon = 0$. Notice that ν_ϕ is equal to $-\infty$ if Player O cannot realize ϕ.

Example 2. Let us come back to Example 1 of a client-server system with two clients sharing a resource. The specification have been formalized by an LTL formula ϕ over the alphabet $P = I \uplus O$, with $I = \{r_1, r_2\}$, $O = \{g_1, w_1, g_2, w_2\}$. Suppose that we want to add the following constraints: client 2's requests take the priority over client 1's requests, but client 1's should still be eventually granted. Moreover, we would like to keep minimal the delay between requests and grants. This latter requirement has more the flavor of an optimality criterion and is best modeled using a weight function and a mean-payoff objective. To this end, we impose penalties to the waiting signals w_1, w_2 controlled by Player O, with a larger penalty to w_2 than to w_1. We thus use the following weight function $w : \mathsf{Lit}(P) \to \mathbb{Z}$:

$$w(l) = \begin{cases} -1 \text{ if } l = w_1 \\ -2 \text{ if } l = w_2 \\ 0 \text{ otherwise.} \end{cases}$$

One optimal strategy for the server is to behave as follows: it almost always grants the resource to client 2 immediately after r_2 is set to true by client 2, and with a decreasing frequency grants request r_1 emitted by client 1. Such a server ensures a mean-payoff value equal to -1 against the most demanding behavior of the clients (where they are constantly requesting the shared resource). Such a strategy requires the server to use an infinite memory as it has to grant client 1 with an infinitely decreasing frequency. Note that a server that would grant client 1 in such a way without the presence of requests by client 1 would still be optimal.

It is easy to see that no finite memory server can be optimal. Indeed, if we allow the server to count only up to a fixed positive integer $k \in \mathbb{N}$ then the best that this server can do is as follows: grant immediatly any request by client 2 if the last ungranted request of client 1 has been emitted less than k steps in the past, otherwise grant the request of client 1. The mean-payoff value of this solution, in the worst-case (when the two clients always emit their respective request) is equal to $-(1 + \frac{1}{k})$. So, even if finite memory cannot be optimal in this example, it is the case that given any $\epsilon > 0$, we can devise a finite-memory strategy that is ϵ-optimal.

LTL_E *realizability and synthesis* – For the proofs of this paper, we also need to consider realizability and synthesis with *energy objectives* (instead of mean-payoff objectives). With the same notations as before, the LTL_E *realizability problem* is to decide whether ϕ is E-*realizable*, that is, whether there exists a strategy λ_O of Player O and an integer $c_0 \in \mathbb{N}$ such that for all strategies λ_I of Player I, *(i)* $u = \mathsf{Outcome}(\lambda_O, \lambda_I) \models \phi$, *(ii)* $\forall n \geq 0 \cdot c_0 + \mathsf{EL}(u(n)) \geq 0$. Instead of requiring that $\mathsf{MP}(u) \geq \nu$ for some given theshold ν as a second objective, we thus ask if there exists an *initial credit* c_0 such that the energy level of each prefix $u(n)$ remains positive. When ϕ is E-realizable, the LTL_E *synthesis problem* is to produce such a winning strategy λ_O for Player O. Finally, we define the *minimum initial credit* as the least value of initial credit for which ϕ is E-realizable. A strategy λ_O is *optimal* if it is winning for the minimum initial credit.

3 Computational complexity of the LTL_MP realizability problem

In this section, we solve the LTL_MP realizability problem, and we establish its complexity. Our solution relies on a reduction to a mean-payoff parity game. The same result also holds for the LTL_E realizability problem.

Theorem 2. *The LTL_MP realizability problem is 2ExpTime-Complete.*

Before proving this result, we recall useful notions on parity automata and on game graphs.

3.1 Deterministic parity automata

A *deterministic parity automaton* over a finite alphabet Σ is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, p)$ where Q is a finite set of states with q_0 the initial state, $\delta : Q \times \Sigma \to Q$ is a transition function that assigns a unique state[4] to each given state and symbol, and $p : Q \to \mathbb{N}$ is a priority function that assigns a priority to each state.

For infinite words $u = \sigma_0\sigma_1 \cdots \in \Sigma^\omega$, there exists a unique run $\rho(u) = \rho_0\rho_1 \cdots \in Q^\omega$ such that $\rho_0 = q_0$ and $\forall k \geq 0 \cdot \rho_{k+1} = \delta(\rho_k, \sigma_k)$. We denote by $\mathsf{Inf}(\rho(u))$ the set of states that appear infinitely often in $\rho(u)$. The *language* $L(\mathcal{A})$ is the set of words $u \in \Sigma^\omega$ such that $\min\{p(q) \mid q \in \mathsf{Inf}(\rho(u))\}$ is even. We have the next theorem (see for instance [22]).

Theorem 3. *Let ϕ be an* LTL *formula over P. One can construct a deterministic parity automaton $\mathcal{A}_\phi$ such that $L(\mathcal{A}_\phi) = [\![\phi]\!]$. If ϕ has size n, then $\mathcal{A}_\phi$ has $2^{2^{O(n \log n)}}$ states and $2^{O(n)}$ priorities.*

3.2 Game graphs

A *game graph* $G = (S, s_0, E)$ consists of a finite set S of states partitioned into S_1 the states of Player 1, and S_2 the states of Player 2 (that is $S = S_1 \uplus S_2$), an initial state s_0, and a set $E \subseteq S \times S$ of edges such that for all $s \in S$, there exists a state $s' \in S$ such that $(s, s') \in E$. A game on G starts from the initial state s_0 and is played in rounds as follows. If the game is in a state belonging to S_1, then Player 1 chooses the successor state among the set of outgoing edges; otherwise Player 2 chooses the successor state. Such a game results in a *play* that is an infinite path $\rho = s_0s_1 \ldots s_n \ldots$, whose prefix $s_0s_1 \ldots s_n$ of length[5] n of is denoted by $\rho(n)$. We denote by $\mathsf{Plays}(G)$ the set of all plays in G and by $\mathsf{Pref}(G)$ the set of all prefixes of plays in G. A *turn-based* game is a game graph G such that $E \subseteq (S_1 \times S_2) \cup (S_2 \times S_1)$, meaning that each game is played in rounds alternatively by Player 1 and Player 2.

Objectives – An *objective* for G is a set $\Omega \subseteq S^\omega$. Let $p : S \to \mathbb{N}$ be a *priority function* and $w : E \to \mathbb{Z}$ be a *weight function* where positive weights represent rewards. The *energy level* of a prefix $\gamma = s_0s_1 \ldots s_n$ of a play is $\mathsf{EL}_G(\gamma) = \sum_{i=0}^{n-1} w(s_i, s_{i+1})$, and the *mean-payoff value* of a play $\rho = s_0s_1 \ldots s_n \ldots$ is $\mathsf{MP}_G(\rho) = \liminf_{n \to \infty} \frac{1}{n} \cdot \mathsf{EL}_G(\rho(n))$.[6] Given a play ρ, we denote $\mathsf{Inf}(\rho)$ the set of states $s \in S$ that appear infinitely often in ρ. The following objectives Ω are considered in the sequel:

– *Safety objective*. Given a set $\alpha \subseteq S$, the safety objective is defined as $\mathsf{Safety}_G(\alpha) = \mathsf{Plays}(G) \cap \alpha^\omega$.
– *Parity objective*. The parity objective is defined as $\mathsf{Parity}_G(p) = \{\rho \in \mathsf{Plays}(G) \mid \min\{p(s) \mid s \in \mathsf{Inf}(\rho)\}$ is even$\}$.
– *Energy objective*. Given an initial credit $c_0 \in \mathbb{N}$, the energy objective is defined as $\mathsf{PosEnergy}_G(c_0) = \{\rho \in \mathsf{Plays}(G) \mid \forall n \geq 0 : c_0 + \mathsf{EL}_G(\rho(n)) \geq 0\}$.
– *Mean-payoff objective*. Given a threshold $\nu \in \mathbb{Q}$, the mean-payoff objective is defined as $\mathsf{MeanPayoff}_G(\nu) = \{\rho \in \mathsf{Plays}(G) \mid \mathsf{MP}_G(\rho) \geq \nu\}$.
– *Combined objective*. The *energy safety* objective $\mathsf{PosEnergy}_G(c_0) \cap \mathsf{Safety}_G(\alpha)$ (resp. *energy parity* objective $\mathsf{PosEnergy}_G(c_0) \cap \mathsf{Parity}_G(p)$, *mean-payoff parity* objective $\mathsf{MeanPayoff}_G(\nu) \cap \mathsf{Parity}_G(p)$) combines the requirements of energy and safety (resp. energy and parity, energy and mean-payoff) objectives.

When an objective Ω is imposed on a game G, we say that G is an Ω *game*. For instance, if Ω is an energy safety objective, we say that G is an *energy safety game*, aso.

---

[4] In this definition, a deterministic parity automaton is also complete
[5] The length is counted as the number of edges.
[6] Notation EL, MP and Outcome is here used with the index G to avoid any confusion with the same notation introduced in the previous section.

Strategies – Given a game graph G, a *strategy* for Player 1 is a function $\lambda_1 : S^* S_1 \rightarrow S$ such that $(s, \lambda_1(\gamma \cdot s)) \in E$ for all $\gamma \in S^*$ and $s \in S_1$. A play $\rho = s_0 s_1 \ldots s_n \ldots$ starting from the initial state s_0 is compatible with λ_1 if for all $k \geq 0$ such that $s_k \in S_1$ we have $s_{k+1} = \lambda_1(\rho(k))$. Strategies and play compatibility are defined symmetrically for Player 2. The set of strategies of Player 1 (resp. Player 2) is denoted by Π_1 (resp. Π_2). We denote by $\mathsf{Outcome}_G(\lambda_1, \lambda_2)$ the play from q_0, called outcome, that is compatible with λ_1 and λ_2. The set of all outcomes $\mathsf{Outcome}_G(\lambda_1, \lambda_2)$, with λ_2 any strategy of Player 2, is denoted by $\mathsf{Outcome}_G(\lambda_1)$. A strategy λ_1 for Player 1 is *winning* for an objective Ω if $\mathsf{Outcome}_G(\lambda_1) \subseteq \Omega$. We also say that λ_1 is winning in the Ω game G.

A strategy λ_1 of Player 1 is *finite-memory* if there exists a right-congruence $\sim$ on $\mathsf{Pref}(G)$ with finite index such that $\lambda_1(\gamma \cdot s_1) = \lambda_1(\gamma' \cdot s_1)$ for all $\gamma \sim \gamma'$ and $s_1 \in S_1$. The *size* of the memory is equal to the number of equivalence classes of $\sim$. We say that λ_1 is *memoryless* if $\sim$ has only one equivalence class. In other words, λ_1 is a mapping $S_1 \rightarrow S$ that only depends on the current state.

Energy safety games – Let us consider a safety game $\langle G, \alpha \rangle$ with the safety objective $\mathsf{Safety}_G(\alpha)$, or equivalently, with the objective to avoid $S \setminus \alpha$. The next classical fixpoint algorithm allows one to check whether Player 1 has a winning strategy (see [19] for example). We define the fixpoint $\mathsf{Win}_1(\alpha)$ of the sequence $W_0 = \alpha$, $W_{k+1} = W_k \cap \{\{s \in S_1 \mid \exists (s, s') \in E \cdot s' \in W_k\} \cup \{s \in S_2 \mid \forall (s, s') \in E \cdot s' \in W_k\}\}$ for all $k \geq 0$. It is well-known that Player 1 has a winning strategy λ_1 in the safety game $\langle G, \alpha \rangle$ iff $s_0 \in \mathsf{Win}_1(\alpha)$, and that the set $\mathsf{Win}_1(\alpha)$ can be computed in polynomial time. Moreover, the subgraph G' of G induced by $\mathsf{Win}_1(\alpha)$ is again a game graph (i.e. every state has an outgoing edge), and if $s_0 \in \mathsf{Win}_1(\alpha)$, then λ_1 can be chosen as a memoryless strategy $S_1 \rightarrow S$ such that $(s, \lambda_1(s))$ is an edge in G' for all $s \in \mathsf{Win}_1(\alpha)$ (Player 1 forces to stay in G'). With this induced subgraph, we have the next reduction of energy safety games to energy games.

Proposition 4 *Let $\langle G, w, \alpha \rangle$ be an energy safety game. Let $\langle G', w' \rangle$ be the energy game such that G' is the subgraph of G induced by $\mathsf{Win}_1(\alpha)$ and w' is the restriction of w to its edges. Then the winning strategies of Player 1 in $\langle G', w' \rangle$ are the winning strategies in $\langle G, w, \alpha \rangle$ that are restricted to the states of $\mathsf{Win}_1(\alpha)$.*
$\square$

For an energy game $\langle G, w \rangle$ (resp. energy safety game $\langle G, w, \alpha \rangle$), the *initial credit problem* asks whether there exist an initial credit $c_0 \in \mathbb{N}$ and a winning strategy for Player 1 for the objective $\mathsf{PosEnergy}_G(c_0)$ (resp. $\mathsf{PosEnergy}_G(c_0) \cap \mathsf{Safety}(\alpha)$). It is known that for energy games, this problem can be solved in $\mathsf{NP} \cap \mathsf{coNP}$, and memoryless strategies suffice to witness the existence of winning strategies for Player 1 [7,11]. Moreover, if we store in the states of the game the current energy level up to some bound $C \geq 0$, one gets a safety game (whose safe states are those states with a positive energy level). For a sufficiently large bound C, this safety game is equivalent to the initial energy game [9]. Intuitively, the states of this safety game are pairs (s, c) with s a state of G and c an energy level in $\mathcal{C} = \{\bot, 0, 1, \ldots C\}$ (with $\bot < 0$). When adding a positive (resp. negative) weight to an energy level, we bound the sum to C (resp. $\bot = -1$). The safety objective is to avoid states of the form $(s, \bot)$. Formally, given $\langle G, w \rangle$ an energy game with $G = (S, s_0, E)$, we define the safety game $\langle G_C, \alpha \rangle$ with a graph $G_C = (S', s'_0, E')$ and a safety objective α as follows:

- $S' = \{(s, c) \mid s \in S, c \in \mathcal{C}\}$
- $s'_0 = (s_0, C)$
- $((s, c), (s', c')) \in E'$ if $e = (s, s') \in E$ and $c \oplus w(e) = c'$
- $\alpha = \{(s, c) \mid s \in S, c \neq \bot\}$

In this definition, we use the operator $\oplus : \mathcal{C} \times \mathbb{Z} \rightarrow \mathcal{C}$ such that $c \oplus k = \min(C, c + k)$ if $\{c \neq \bot$ and $c + k \geq 0\}$, and $\bot$ otherwise.

By Proposition 4, it follows that energy safety games can be reduced to safety games.

Theorem 5. *Let $\langle G, w, \alpha \rangle$ be an energy safety game. Then one can construct a safety game $\langle G', \alpha' \rangle$ such that Player 1 has a winning strategy in $\langle G, w, \alpha \rangle$ iff he has a winning strategy in $\langle G', \alpha' \rangle$.*
$\square$

Energy parity games and mean-payoff parity games – Given an energy parity game $\langle G, w, p \rangle$, we can also formulate the *initial credit problem* as done previously for energy games.

Theorem 6 ([12]). *The initial credit problem for a given energy parity game $\langle G, w, p \rangle$ can be solved in time $O(|E| \cdot d \cdot |S|^{d+3} \cdot W)$ where $|E|$ is the number of edges of G, d is the number of priorities used by p and W is the largest weight (in absolute value) used by w. Moreover if Player 1 wins, then he has a finite-memory winning strategy with a memory size bounded by $4 \cdot |S| \cdot d \cdot W$.*

Let us turn to mean-payoff parity games $\langle G, w, p \rangle$. With each play $\rho \in \mathsf{Plays}(G)$, we associate a *value* $\mathsf{Val}_G(\rho)$ defined as follows (as done in the context of $\mathsf{LTL}_{\mathsf{MP}}$ realizability):

$$\mathsf{Val}_G(\rho) = \begin{cases} \mathsf{MP}_G(\rho) & \text{if } \rho \in \mathsf{Parity}_G(p) \\ -\infty & \text{otherwise.} \end{cases}$$

The *optimal value* for Player 1 is defined as

$$\nu_G = \sup_{\lambda_1 \in \Pi_1} \inf_{\lambda_2 \in \Pi_2} \mathsf{Val}_G(\mathsf{Outcome}_G(\lambda_1, \lambda_2)).$$

For a real-valued $\epsilon \geq 0$, a strategy λ_1 for Player 1 is ϵ-*optimal* if $\mathsf{Val}_G(\mathsf{Outcome}_G(\lambda_1, \lambda_2)) \geq \nu_G - \epsilon$ against all strategies λ_2 of Player 2. It is *optimal* if it is ϵ-optimal with $\epsilon = 0$. If Player 1 cannot achieve the parity objective, then $\nu_G = -\infty$, otherwise optimal strategies exist [14] and ν_G is the largest threshold ν for which Player 1 can hope to achieve $\mathsf{MeanPayoff}_G(\nu)$.

Theorem 7 ([14,8,15]). *The optimal value of a mean-payoff parity game $\langle G, w, p \rangle$ can be computed in time $O(|E| \cdot |S|^{d+2} \cdot W)$. When $\nu_G \neq -\infty$, optimal strategies for Player 1 may require infinite memory; however for all $\epsilon > 0$ Player 1 has a finite-memory ϵ-optimal strategy.*

3.3 Reduction to a mean-payoff parity game

Solution to the $\mathsf{LTL}_{\mathsf{MP}}$ realizability problem – We can now proceed to the proof of Theorem 2. It is based on the following proposition.

Proposition 8 *Let ϕ be an LTL formula over $\langle P, w_P \rangle$. Then one can construct a mean-payoff parity game $\langle G_\phi, w, p \rangle$ with $2^{2^{O(n \log n)}}$ states and $2^{O(n)}$ priorities such that the following are equivalent: for each threshold $\nu \in \mathbb{Q}$*

1. *there exists a (finite-memory) strategy λ_O of Player O such that $\mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I)) \geq \nu$ against all strategies λ_I of Player I;*
2. *there exists a (finite-memory) strategy λ_1 of Player 1 such that $\mathsf{Val}_{G_\phi}(\mathsf{Outcome}_{G_\phi}(\lambda_1, \lambda_2)) \geq \nu$ against all strategies λ_2 of Player 2, in the game $\langle G_\phi, w, p \rangle$.*

Moreover, if λ_1 is a finite-memory strategy with size m, then λ_O can be chosen as a finite-memory strategy with size $m \cdot |S_1|$ where S_1 is the set of states of Player 1 in G_ϕ.[7]

Proof. Let ϕ be an LTL formula over $P = I \uplus O$, and let $w_P : \mathsf{Lit}(P) \to \mathbb{Z}$ be a weight function. We first construct a deterministic parity automaton $\mathcal{A}_\phi = (\Sigma_P, Q, q_0, \delta, p)$ such that $L(\mathcal{A}_\phi) = [\![\phi]\!]$ (see Theorem 3). This automaton has $2^{2^{O(n \log n)}}$ states and $2^{O(n)}$ priorities.

We then derive from $\mathcal{A}_\phi$ a turn-based mean-payoff parity game $\langle G_\phi, w, p' \rangle$ with $G_\phi = (S, s_0, E)$ as follows. The initial state s_0 is equal to (q_0, j_0) for some $j_0 \in I$.[8] To get the turn-based aspect, the set S is partitioned as $S_1 \uplus S_2$ such that $S_1 = \{(q, i) \mid q \in Q, i \in \Sigma_I\}$ and $S_2 = \{(q, o) \mid q \in Q, o \in \Sigma_O\}$.[9] Let us describe the edges of G_ϕ. For each $q \in Q$, $o \in \Sigma_O$ and $i \in \Sigma_I$, let $q' = \delta(q, o \cup i)$. Then E contains the two edges $((q, j), (q, o))$ and $((q, o), (q', i))$ for all $j \in \Sigma_I$. We clearly have $E \subseteq (S_1 \times S_2) \cup (S_2 \times S_1)$.

---

[7] A converse of this corollary could also be stated but is of no utility in the next results.

[8] Symbol j_0 can be chosen arbitrarily.

[9] We will limit the set of states to the accessible ones.

Moreover, since $\mathcal{A}_\phi$ is deterministic, we have the nice property that there exists a bijection $\Theta : \Sigma_P^* \to$ $\mathsf{Pref}(G_\phi) \cap (S_1 S_2)^* S_1$ defined as follows. For each $u = (o_0 \cup i_0)(o_1 \cup i_1) \dots (o_n \cup i_n) \in \Sigma_P^*$, we consider in $\mathcal{A}_\phi$ the run $\rho(u) = q_0 q_1 \dots q_{n+1} \in Q^*$ such that $q_{k+1} = \delta(q_k, o_k \cup i_k)$ for all k. We then define $\Theta(u)$ as $(q_0, j_0)(q_0, o_0)(q_1, i_0)(q_1, o_1)(q_2, i_1) \cdots (q_n, o_n)(q_{n+1}, i_n)$. Clearly, Θ is a bijection and it can be extended to a bijection $\Theta : \Sigma_P^\omega \to \mathsf{Plays}(G_\phi)$.

The priority function $p' : S \to \mathbb{N}$ for G_ϕ is defined from the priority function p of $\mathcal{A}_\phi$ by $p'(q, o) = p'(q, i) = p(q)$ for all $q \in Q$, $o \in O$ and $i \in I$. The weight function $w : E \to \mathbb{Z}$ for G_ϕ is defined from the weight function w_P as follows. For all edges e ending in a state (q, o) with $o \in O$ (resp. (q, i) with $i \in I$), we define $w(e) = w_P(o)$ (resp. $w(e) = w_P(i)$). Notice that Θ preserves the energy level, the meanpayoff value and the parity objective, since for each $u \in \Sigma_P^\omega$, we have *(i)* $\mathsf{EL}(u(n)) = \mathsf{EL}_{G_\phi}(\Theta(u(n)))$ for all n[10], *(ii)* $\mathsf{MP}(u) = \mathsf{MP}_{G_\phi}(\Theta(u))$, and *(iii)* $u \models \phi$ iff $\Theta(u)$ satisfies the objective $\mathsf{Parity}_{G_\phi}(p')$.

It is now easy to prove that the two statements of Proposition 8 are equivalent. Suppose that *1.* holds. Given the winning strategy $\lambda_O : (\Sigma_O \Sigma_I)^* \to \Sigma_O$, we are going to define a winning strategy $\lambda_1 : (S_1 S_2)^* S_1 \to S_2$ of Player 1 in G_ϕ, with the idea that λ_1 mimics λ_O thanks to the bijection Θ. More precisely, for any prefix $\gamma \in (S_1 S_2)^* S_1$ compatible with λ_1, we let $\lambda_1(\gamma) = (q, o)$ such that (q, i_n) is the last state of γ and $o = \lambda_O(o_0 i_0 o_1 i_1 \dots o_n i_n)$ with $(o_0 \cup i_0)(o_1 \cup i_1) \dots (o_n \cup i_n) = \Theta^{-1}(\gamma)$. In this way, λ_1 is winning. Indeed $\mathsf{Val}_{G_\phi}(\mathsf{Outcome}_{G_\phi}(\lambda_1, \lambda_2)) \geq \nu$ for any strategy λ_2 of Player 2 because λ_O is winning and by *(ii)* and *(iii)* above. Moreover λ_O is finite-memory iff λ_1 is finite-memory.

Suppose now that *2.* holds. Given the (finite-memory) winning strategy λ_1, we define a (finite-memory) winning strategy λ_O with the same kind of arguments as done above. We just give the definition of λ_O from λ_1. We let $\lambda_O(o_0 i_0 o_1 i_1 \dots o_n i_n) = o$ such that $\lambda_1(\gamma) = (q, o)$ with $\gamma = \Theta((o_0 \cup i_0)(o_1 \cup i_1) \dots (o_n \cup i_n))$.

Suppose now that λ_1 is finite-memory with size m. Let $\sim_1$ be a right-congruence on $\mathsf{Pref}(G_\phi)$ with index m such that $\lambda_1(\gamma \cdot s) = \lambda_1(\gamma' \cdot s)$ for all $\gamma \sim_1 \gamma'$ and $s \in S_1$. To show that λ_O is finite-memory, we have to define a right-congruence $\sim_O$ on $(\Sigma_O \Sigma_I)^*$ with finite index such that $\lambda_O(u) = \lambda_O(u')$ for all $u \sim_O u'$. Let $u = o_0 i_0 \dots o_n i_n, u' = o_0' i_0' \dots o_l' i_l' \in (\Sigma_O \Sigma_I)^*$, let $\Theta((o_0 \cup i_0) \dots (o_n \cup i_n)) = \gamma \cdot s, \Theta((o_0' \cup i_0') \dots (o_l' \cup i_l')) = \gamma' \cdot s'$. Looking at the definition of λ_O, we see that $\sim_O$ has to be defined such that $u \sim_O u'$ if $\gamma \sim_1 \gamma'$ and $s = s'$. Moreover $\sim_O$ has index $m \cdot |S_1|$.

This completes the proof.[11] $\qquad\square$

Proof (of Theorem 2). By Proposition 8, solving the $\mathsf{LTL}_{\mathsf{MP}}$ realizability problem is equivalent to checking whether Player 1 has a winning strategy in the mean-payoff parity game $\langle G_\phi, w, p \rangle$ for the given threshold ν. By Theorem 7, this check can be done in time $O(|E| \cdot |S|^{d+2} \cdot W)$. Since G_ϕ has $2^{2^{O(n \log n)}}$ states and $2^{O(n)}$ priorities (see Proposition 8), the $\mathsf{LTL}_{\mathsf{MP}}$ realizability problem is in $O(2^{2^{O(n \log n)}} W)$.

This proves the 2Exptime-easyness of $\mathsf{LTL}_{\mathsf{MP}}$ realizability problem. The 2Exptime-hardness of this problem is a consequence of the 2Exptime-hardness of LTL realizability problem (see Theorem 1). $\qquad\square$

Proposition 8 and its proof lead to the next two interesting corollaries. The first corollary is immediate. The second one asserts that the optimal value can be approached with finite memory strategies.

Corollary 9 *Let ϕ be an LTL formula and $\langle G_\phi, w, p \rangle$ be the associated mean-payoff parity game graph. Then $\nu_\phi = \nu_{G_\phi}$. Moreover, when $\nu_\phi \neq -\infty$, one can construct an optimal strategy λ_1 for Player 1 from an optimal strategy λ_O for Player O, and conversely.* $\qquad\square$

Corollary 10 *Let ϕ be an LTL formula. If ϕ is MP-realizable, then for all $\epsilon > 0$, Player O has an ϵ-optimal winning strategy that is finite-memory, that is*

$$\nu_\phi = \sup_{\substack{\lambda_O \in \Pi_O \\ \lambda_O \text{ finite-memory}}} \inf_{\lambda_I \in \Pi_I} \mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I)).$$

Proof. Suppose that ϕ is MP-realizable. By Corollary 9, $\nu_\phi = \nu_{G_\phi} \neq -\infty$. Therefore, by Proposition 8 and Theorem 7, for each $\epsilon > 0$, Player 1 has a finite-memory winning strategy λ_1 in the mean-payoff

---

[10] For this equality, it is useful to recall footnote 3.

[11] Notice that we could have defined a smaller game graph G_ϕ with $S_1 = Q$ and $w((q, o), q') = \max\{w_P(i) \mid \delta(q, o \cup i) = q'\}$ for all $(q, o) \in S_2, q' \in S_1$. The current definition simplifies the proof.

parity $G_\phi = (S, s_0, E)$ for the threshold $\nu_\phi - \epsilon$. By Proposition 8, from λ_1, we can derive a finite-memory winning strategy λ_O for the $\mathsf{LTL_{MP}}$ realizability of ϕ for this threshold, which is thus the required finite-memory ϵ-optimal winning strategy. □

Solution to the $\mathsf{LTL_E}$ realizability problem – The same kind of arguments show that the $\mathsf{LTL_E}$ realizability problem is 2Exptime-complete. Indeed, in Proposition 8, it is enough to use a reduction with the same game $\langle G_\phi, w, p \rangle$, however with energy parity objectives instead of mean-payoff parity objectives. The proof is almost identical by taking the same initial credit c_0 for both the $\mathsf{LTL_E}$ realizability of ϕ and the energy objective in G_ϕ, and by using Theorem 6 instead of Theorem 7.

Theorem 11. *The $\mathsf{LTL_E}$ realizability problem is 2ExpTime-Complete.* □

The next proposition states that when ϕ is E-realizable, then Player O has a finite-memory strategy the size of which is related to G_ϕ. This result is stronger than Corollary 10 since it also holds for optimal strategies and it gives a bound on the memory size of the winning strategy.

Proposition 12 *Let ϕ be an LTL formula over $\langle P, w_P \rangle$ and $\langle G_\phi, w, p \rangle$ be the associated energy parity game. Then ϕ is E-realizable iff it is E-realizable under finite memory. Moreover Player O has a finite-memory winning strategy with a memory size bounded by $4 \cdot |S|^2 \cdot d \cdot W$, where $|S|$ is the number of states of G_ϕ, d its number of priorities and W its largest absolute weight.*

Proof. By Proposition 8 where $G_\phi = (S, s_0, E)$ is considered as an energy parity game, we know that Player 1 has a winning strategy λ_1 for the initial credit problem in G_ϕ. Moreover, by Theorem 6, we can suppose that this strategy has finite-memory M with $|M| \le 4 \cdot |S| \cdot d \cdot W$. Finally, one can derive a finite-memory winning strategy λ_O for the $\mathsf{LTL_E}$ realizabilty of ϕ with a memory size bounded by $|M| \cdot |S|$ by Proposition 8. □

4 Safraless algorithm

In the previous section, we have proposed an algorithm for solving the $\mathsf{LTL_{MP}}$ realizability of a given LTL formula ϕ, which is based on a reduction to the mean-payoff parity game G_ϕ. This algorithm has two main drawbacks. First, it requires the use of Safra's construction to get a deterministic parity automaton $\mathcal{A}_\phi$ such that $L(\mathcal{A}_\phi) = [\![\phi]\!]$. This construction is intricate and notoriously difficult to implement efficiently [1]. Second, strategies for the game G_ϕ may require infinite memory (for the threshold ν_{G_ϕ}, see Theorem 7). This can also be the case for the $\mathsf{LTL_{MP}}$ realizability problem, as illustrated by Example 2. In this section, we show how to circumvent these two drawbacks.

4.1 Finite-memory strategies

The second drawback (infinite memory strategies) has been already partially solved by Corollary 10, when the threshold given for the $\mathsf{LTL_{MP}}$-realizability is the optimal value ν_ϕ. Indeed it states the existence of finite-memory winning strategies for the thresholds $\nu_\phi - \epsilon$, for all $\epsilon > 0$. We here show that we can go further by translating the $\mathsf{LTL_{MP}}$ realizability problem under finite memory into an $\mathsf{LTL_E}$ realizability problem, and conversely.

Recall (see Proposition 8) that testing whether an LTL formula ϕ is MP-realizable for a given threshold ν is equivalent to solve the mean-payoff parity game G_ϕ for the same threshold. Moreover, with the same game graph, testing whether ϕ is E-realizable is equivalent to solve the energy parity game G_ϕ. Let us study in more details the finite-memory winning strategies of G_ϕ seen either as a mean-payoff parity game, or as an energy parity game, through the next property proved in [12].

Proposition 13 ([12]) *Let $G = (S, s_0, E)$ be a game with a priority function p and a weight function w. Let ν be a threshold and $w - \nu$ be the weight function such that $(w - \nu)(e) = w(e) - \nu$ for all edges e of G. Let λ_1 be a finite-memory strategy for Player 1. Then λ_1 is winning in the mean-payoff parity game $\langle G, w, p \rangle$ with threshold ν iff λ_1 is winning in the energy parity game $\langle G, w - \nu, p \rangle$ for some initial credit c_0.*

This proposition leads to the next theorem.

Theorem 14. *Let ϕ be an* LTL *formula ϕ over $\langle P, w_P \rangle$, and $\langle G_\phi, w, p \rangle$ be its associated mean-payoff parity game. Then*

- *the formula ϕ is* MP-*realizable under finite memory for threshold ν iff ϕ over $\langle P, w_P - \nu \rangle$ is* E-*realizable*
- *if ϕ is* MP-*realizable under finite memory, Player O has a winning strategy whose memory size is bounded by $4 \cdot |S|^2 \cdot d \cdot W$, where $|S|$ is the number of states of G_ϕ, d is the number of priorities of p and W is the largest absolute weight of the weight function $w - \nu$.*

Proof. If ϕ is MP-realizable under finite-memory for threshold ν, then by Proposition 8, Player 1 has a finite-memory strategy λ_1 in the mean-payoff parity game $\langle G_\phi, w, p \rangle$. By Proposition 13, λ_1 is a winning strategy in the energy parity game $\langle G_\phi, w - \nu, p \rangle$, that shows (by Proposition 8) that ϕ is LTL$_E$-realizable with weight function $w_P - \nu$ over Lit(P). The converse is proved similarly.

Finally, given a winning strategy for the LTL$_E$-realizability of ϕ with weight function $w - \nu$ over Lit(P), we can suppose by Proposition 12 that it is finite-memory with a memory size bounded by $4 \cdot |S|^2 \cdot d \cdot W$, where S, d and W are the parameters of the energy parity game $\langle G_\phi, w - \nu, p \rangle$. This concludes the proof.
□

The next corollary follows from Theorem 14 and Corollary 10.

Corollary 15 *Let ϕ be an* LTL *formula over $\langle P, w_P \rangle$. Then for all $\epsilon \in \mathbb{Q}$, with $\epsilon > 0$, the following are equivalent:*

1. *λ_O is a finite-memory ϵ-optimal winning strategy for the* LTL$_{MP}$-*realizability of ϕ*
2. *λ_O is a winning strategy for the* LTL$_E$-*realizability of ϕ with weight function $w_P - \nu_\phi + \epsilon$ over* Lit(P).

It is important to notice that in this corollary, the memory size of the strategy (as described in Theorem 14) increases as ϵ decreases. Indeed, it depends on the weight function $w - \nu_\phi + \epsilon$ used by the energy parity game G_ϕ. We recall that if $\epsilon = \frac{a}{b}$, then this weight function must be multiplied by b in a way to have integer weights (see footnote 2). The largest absolute weight W is thus also multiplied by b.

In the sequel, to avoid strategies with infinite memory when solving the LTL$_{MP}$ realizability problem, we will restrict to the LTL$_{MP}$ realizability problem under finite memory. By Theorem 14, it is enough to study winning strategies for the LTL$_E$ realizability problem (having in mind that the weight function over Lit(P) has to be adapted). In the sequel, we only study this problem.

4.2 Safraless construction

To avoid the Safra's construction needed to obtain a deterministic parity automaton for the underlying LTL formula, we adapt a Safraless construction proposed in [18,24] for the LTL synthesis problem, in a way to deal with weights and efficiently solve the LTL$_E$ synthesis problem. Instead of constructing a mean-payoff parity game from a deterministic parity automaton as done in Proposition 8, we will propose a reduction to a safety game. In this aim, we need to define the notion of energy automaton.

Energy automata – Let $\langle P, w \rangle$ with P a finite set of signals and w a weight function over Lit(P). We are going to recall several notions of automata on infinite words over Σ_P and introduce the related notion of energy automata over the weighted alphabet $\langle \Sigma_P, w \rangle$. An *automaton* $\mathcal{A}$ over the alphabet Σ_P is a tuple $(\Sigma_P, Q, q_0, \alpha, \delta)$ such that Q is a finite set of states, $q_0 \in Q$ is the initial state, $\alpha \subseteq Q$ is a set of final states and $\delta : Q \times \Sigma_P \to 2^Q$ is a transition function. We say that $\mathcal{A}$ is *deterministic* if $\forall q \in Q \cdot \forall \sigma \in \Sigma_P \cdot |\delta(q, \sigma)| \leq 1$. It is *complete* if $\forall q \in Q \cdot \forall \sigma \in \Sigma_P \cdot \delta(q, \sigma) \neq \varnothing$.

A *run* of $\mathcal{A}$ on a word $u = \sigma_0 \sigma_1 \cdots \in \Sigma_P^\omega$ is an infinite sequence of states $\rho = \rho_0 \rho_1 \cdots \in Q^\omega$ such that $\rho_0 = q_0$ and $\forall k \geq 0 \cdot \rho_{k+1} \in \delta(\rho_k, \sigma_k)$. We denote by Runs$_\mathcal{A}(u)$ the set of runs of $\mathcal{A}$ on u, and by Visit(ρ, q) the number of times the state q occurs along the run ρ. We consider the following acceptance conditions:

$$\begin{array}{ll} \text{\textit{Non-deterministic Büchi}:} & \exists \rho \in \mathsf{Runs}_{\mathcal{A}}(u) \cdot \exists q \in \alpha \cdot \mathsf{Visit}(\rho, q) = \infty \\ \text{\textit{Universal co-Büchi}:} & \forall \rho \in \mathsf{Runs}_{\mathcal{A}}(u) \cdot \forall q \in \alpha \cdot \mathsf{Visit}(\rho, q) < \infty \\ \text{\textit{Universal K-co-Büchi}:} & \forall \rho \in \mathsf{Runs}_{\mathcal{A}}(u) \cdot \sum_{q \in \alpha} \mathsf{Visit}(\rho, q) \leq K. \end{array}$$

A word $u \in \Sigma_P^\omega$ is *accepted* by a *non-deterministic Büchi automaton* (NB) $\mathcal{A}$ if u satisfies the non-deterministic Büchi acceptance condition. We denote by $L_{\mathrm{nb}}(\mathcal{A})$ the set of words accepted by $\mathcal{A}$. Similarly we have the notion of *universal co-Büchi automaton* (UCB) $\mathcal{A}$ (resp. *universal K-co-Büchi automaton* (UKCB) $\langle \mathcal{A}, K \rangle$) and the set $L_{\mathrm{ucb}}(\mathcal{A})$ (resp. $L_{\mathrm{ucb}, K}(\mathcal{A})$) of accepted words.

We also now introduce the energy automata. Let $\mathcal{A}$ be a NB over the alphabet Σ_P. The related *energy non-deterministic Büchi automaton* (eNB) $\mathcal{A}^w$ is over the weighted alphabet $\langle \Sigma_P, w \rangle$ and has the same structure as $\mathcal{A}$. Given an initial credit $c_0 \in \mathbb{N}$, a word u is *accepted* by $\mathcal{A}^w$ if *(i)* u satisfies the non-deterministic Büchi acceptance condition and *(ii)* $\forall n \geq 0 \cdot c_0 + \mathsf{EL}(u(n)) \geq 0$. We denote by $L_{\mathrm{nb}}(\mathcal{A}^w, c_0)$ the set of words accepted by $\mathcal{A}^w$ with the given initial credit c_0. We also have the notions of *energy universal co-Büchi automaton* (eUCB) $\mathcal{A}^w$ and *energy universal K-co-Büchi automaton* (eUKCB) $\langle \mathcal{A}^w, K \rangle$, and the related sets $L_{\mathrm{ucb}}(\mathcal{A}^w, c_0)$ and $L_{\mathrm{ucb}, K}(\mathcal{A}^w, c_0)$. Notice that if $K \leq K'$, then $L_{\mathrm{ucb}, K}(\mathcal{A}^w, c_0) \subseteq L_{\mathrm{ucb}, K'}(\mathcal{A}^w, c_0)$, and that if $c_0 \leq c_0'$, then $L_{\mathrm{ucb}, K}(\mathcal{A}^w, c_0) \subseteq L_{\mathrm{ucb}, K}(\mathcal{A}^w, c_0')$.

The interest of UKCB is that they can be determinized with the subset construction extended with counters [18,24]. This construction also holds for eUKCB. Intuitively, for all states q of $\mathcal{A}$, we count (up to $\top = K + 1$) the maximal number of accepting states which have been visited by runs ending in q. This counter is equal to -1 when no run ends in q. The final states are the subsets in which a state has its counter greater than K (accepted runs will avoid them). Formally, let $\mathcal{A}^w$ be a UKCB ($\langle \Sigma_P, w \rangle, Q, q_0, \alpha, \delta$) with $K \in \mathbb{N}$. With $\mathcal{K} = \{-1, 0 \ldots, K, \top\}$ (with $\top > K$), we define $\det(\mathcal{A}^w, K) = (\langle \Sigma_P, w \rangle, \mathcal{F}, F_0, \beta, \Delta)$ where:

- $\mathcal{F} = \{F \mid F \text{ is a mapping from } Q \text{ to } \mathcal{K}\}$
- $F_0 = q \in Q \mapsto \begin{cases} -1 & \text{if } q \neq q_0 \\ (q_0 \in \alpha) & \text{otherwise} \end{cases}$
- $\beta = \{F \in \mathcal{F} \mid \exists q, F(q) = \top\}$
- $\Delta(F, \sigma) = q \mapsto \max\{F(p) \oplus (q \in \alpha) \mid q \in \delta(p, \sigma)\}$

In this definition, $(q \in \alpha) = 1$ if q is in α, and 0 otherwise; we use the operator $\oplus : \mathcal{K} \times \{0, 1\} \to \mathcal{K}$ such that $k \oplus b = -1$ if $k = -1$, $k \oplus b = k + b$ if $\{k \neq -1, \top \text{ and } k + b \leq K\}$, and $k \oplus b = \top$ in all other cases. The automaton $\det(\mathcal{A}^w, K)$ has the next properties:

Proposition 16 *Let $K \in \mathbb{N}$ and $\langle \mathcal{A}^w, K \rangle$ be an eUKCB. Then $\det(\mathcal{A}^w, K)$ is a deterministic and complete energy automaton such that $L_{\mathrm{ucb}, 0}(\det(\mathcal{A}^w, K), c_0) = L_{\mathrm{ucb}, K}(\mathcal{A}^w, c_0)$ for all $c_0 \in \mathbb{N}$.* □

Energy UKCB *and* LTL$_\mathsf{E}$ *realizability* – We now go through a series of results in a way to construct a safety game from a UCB $\mathcal{A}$ such that $L_{\mathrm{ucb}}(\mathcal{A}) = \llbracket \phi \rrbracket$ (see Theorem 19).

Proposition 17 *Let ϕ be an* LTL *formula over $\langle P, w \rangle$. Then there exists a* UCB $\mathcal{A}$ *such that $L_{ucb}(\mathcal{A}) = \llbracket \phi \rrbracket$. Moreover, with the related* eUCB $\mathcal{A}^w$, *the formula ϕ is* E*-realizable with the initial credit c_0 iff there exists a Moore machine $\mathcal{M}$ such that $L(\mathcal{M}) \subseteq L_{ucb}(\mathcal{A}^w, c_0)$.*

Proof. Let us take the negation $\neg \phi$ of ϕ. It is well known (see for instance [22]) that there exists a NB $\mathcal{A}$ such that $L_{\mathrm{nb}}(\mathcal{A}) = \llbracket \neg \phi \rrbracket$; moreover $\overline{L_{\mathrm{nb}}(\mathcal{A})} = L_{\mathrm{ucb}}(\mathcal{A})$ as the accepting conditions are dual. In this way we get the first part of the proposition. The second part follows from Proposition 12 and the definition of Moore machines. □

Theorem 18. *Let ϕ be an* LTL *formula over $\langle P, w_P \rangle$. Let $\langle G_\phi, w, p \rangle$ be the associated energy parity game with $|S|$ be its the number of states, d its number of priorities and W its largest absolute weight. Let $\mathcal{A}$ be a* UCB *with n states such that $L_{ucb}(\mathcal{A}) = \llbracket \phi \rrbracket$. Let $\mathbb{K} = 4 \cdot n \cdot |S|^2 \cdot d \cdot W$ and $\mathbb{C} = \mathbb{K} \cdot W$. Then ϕ is* E*-realizable iff there exists a Moore machine $\mathcal{M}$ such that $L(\mathcal{M}) \subseteq L_{ucb, \mathbb{K}}(\mathcal{A}^w, \mathbb{C})$.*

Proof. By Propositions 12 and 17, ϕ is E-realizable for some initial credit c_0 iff there exists a Moore machine $\mathcal{M}$ such that $L(\mathcal{M}) \subseteq L_{\text{ucb}}(\mathcal{A}^w, c_0)$ and $|\mathcal{M}| = 4 \cdot |S|^2 \cdot d \cdot W$. Consider now the product of $\mathcal{M}$ and $\mathcal{A}^w$: in any accessible cycle of this product, there is no accepting state of $\mathcal{A}^w$ (as shown similarly for the qualitative case [18]) and the sum of the weights must be positive. The length of a path reaching such a cycle is at most $n \cdot |\mathcal{M}|$, therefore one gets $L(\mathcal{M}) \subseteq L_{\text{ucb}, n \cdot |\mathcal{M}|}(\mathcal{A}^w, n \cdot |\mathcal{M}| \cdot W)$. $\qquad\square$

Theorem 19. *Let ϕ be an* LTL *formula. Then one can construct a safety game in which Player 1 has a winning strategy iff ϕ is* E-*realizable.*

Proof. Given ϕ an LTL formula, let us describe the structure of the announced safety game $\langle G'_{\phi, \mathbb{K}, \mathbb{C}}, \alpha' \rangle$. The involved constants $\mathbb{K}$ and $\mathbb{C}$ are those of Theorem 18. By Theorem 5, it is enough to show the statement with an energy safety game instead of a safety game. The construction of this energy safety game is very similar to the construction of a mean-payoff parity game from a deterministic parity automaton as given in the proof of Proposition 8. The main difference is that we will here use a $\text{U}K\text{CB}$ instead of a parity automaton.

Let ϕ be an LTL formula and $\mathcal{A}$ be a UCB such that $L_{\text{ucb}}(\mathcal{A}) = [\![\phi]\!]$. By Theorem 18 and Proposition 16, ϕ is E-realizable iff there exists a Moore machine $\mathcal{M}$ such that $L(\mathcal{M}) \subseteq L_{\text{ucb}, 0}(\det(\mathcal{A}^w, \mathbb{K}), \mathbb{C})$. Exactly as in Proposition 8, we derive from $\det(\mathcal{A}^w, \mathbb{K}) = (\langle \Sigma_P, w \rangle, \mathcal{F}, F_0, \beta, \Delta)$ a turn-based energy safety game $\langle G'_{\phi, \mathbb{K}}, w, \alpha' \rangle$ as follows. The construction of the graph and the definition of w are the same, and $\alpha' = \mathcal{F} \setminus \beta$. Similarly we have a bijection $\Theta : \Sigma_P^* \to \text{Pref}(G'_{\phi, \mathbb{K}}) \cap (S_1 S_2)^* S_1$ that can be extended to a bijection $\Theta : \Sigma_P^\omega \to \text{Plays}(G'_{\phi, \mathbb{K}})$. One can verify that for each $u \in \Sigma_P^\omega$, we have *(i)* $\text{EL}(u(n)) = \text{EL}_{G'_{\phi, \mathbb{K}}}(\Theta(u(n)))$ for all $n \geq 0$, and *(ii)* $\sum_{q \in \beta} \text{Visit}(\rho, q) = 0$ for all runs ρ on u iff $\Theta(u)$ satisfies the objective $\text{Safety}_{G'_{\phi, \mathbb{K}}}(\alpha')$. It follows that $u \in L_{\text{ucb}, 0}(\det(\mathcal{A}^w, \mathbb{K}), \mathbb{C})$ iff $\Theta(u)$ satisfies the combined objective $\text{Safety}_{G'_{\phi, \mathbb{K}}}(\alpha') \cap \text{PosEnergy}_{G'_{\phi, \mathbb{K}}}(\mathbb{C})$ (*).

Suppose that ϕ is E-realizable. By Theorem 18, there exists a finite-memory strategy λ_O represented by a Moore machine $\mathcal{M}$ as given before. As in the proof of Proposition 8, we use Θ to derive a strategy λ_1 of Player 1 that mimics λ_O. As $L(\mathcal{M}) \subseteq L_{\text{ucb}, 0}(\det(\mathcal{A}^w, \mathbb{K}), \mathbb{C})$, by (*), it follows that λ_1 is winning in the energy safety game $\langle G'_{\phi, \mathbb{K}}, w, \alpha' \rangle$ with the initial credit $\mathbb{C}$.

Conversely, suppose now that λ_1 is a winning strategy in $\langle G'_{\phi, \mathbb{K}}, w, \alpha' \rangle$ with the initial credit $\mathbb{C}$. We again use Θ to derive a strategy $\lambda_O : (\Sigma_O \Sigma_I)^* \to \Sigma_O$ that mimics λ_1. As λ_1 is winning, by (*), we have $\text{outcome}(\lambda_O) \subseteq L_{\text{ucb}, 0}(\det(\mathcal{A}^w, \mathbb{K}), \mathbb{C})$. It follows that ϕ is E-realizable. $\qquad\square$

A careful analysis of the complexity of the proposed Safraless procedure shows that it is in 2ExpTime.

5 Antichains

In Section 4, we have shown how to reduce the LTL_E realizability problem to a safety game. In this section we explain how to efficiently and symbolically solve this safety game with antichains.

5.1 Description of the safety game

In the proof of Theorem 19, we have shown how to construct a safety game $\langle G'_{\phi, \mathbb{K}, \mathbb{C}}, \alpha' \rangle$ from an LTL formula ϕ such that ϕ is E-realizable iff Player 1 has a winning strategy in this game. Let us give the precise construction of this game, but more generally for any values $K, C \in \mathbb{N}$. Let $\mathcal{A} = (\langle \Sigma_P, w \rangle, Q, q_0, \alpha, \delta)$ be a UCB such that $L_{\text{ucb}}(\mathcal{A}) = [\![\phi]\!]$, and $\det(\mathcal{A}^w, K) = (\langle \Sigma_P, w \rangle, \mathcal{F}, F_0, \beta, \Delta)$ be the related energy deterministic automaton. Let $\mathcal{C} = \{\bot, 0, 1, \ldots, C\}$. The turned-based safety game $\langle G'_{\phi, K, C}, \alpha' \rangle$ with $G'_{\phi, K, C} = (S = S_1 \uplus S_2, s_0, E)$ has the following structure:

- $S_1 = \{(F, i, c) \mid F \in \mathcal{F}, i \in \Sigma_I, c \in \mathcal{C}\}$
- $S_2 = \{(F, o, c) \mid F \in \mathcal{F}, o \in \Sigma_O, c \in \mathcal{C}\}$
- $s_0 = (F_0, j_0, C)$ with j_0 be an arbitrary symbol of Σ_I, and F_0 be the initial state of $\det(\mathcal{A}^w, \mathbb{K})$
- For all $\Delta(F, o \cup i) = F'$, $j \in \Sigma_I$ and $c \in \mathcal{C}$, the set E contains the edges

$$((F, j, c), (F, o, c')) \text{ and } ((F, o, c'), (F', i, c''))$$

where $c' = c \oplus w(o)$ and $c'' = c' \oplus w(i)$

$$- \; \alpha' = (S_1 \uplus S_2) \setminus \{(F, \sigma, c) \mid \exists q \cdot F(q) = \top \text{ or } c = \bot\}$$

Notice that given $K_1 \leq K_2$ and $C_1 \leq C_2$, if Player 1 has a winning strategy in the safety game $\langle G'_{\phi, K_1, C_1}, \alpha' \rangle$, then he has a winning strategy in the safety game $\langle G'_{\phi, K_2, C_2}, \alpha' \rangle$. The next corollary of Theorem 19 holds.

Corollary 20 *Let ϕ be an* LTL *formula, and $K, C \in \mathbb{N}$. If Player 1 has a winning strategy in the safety game $\langle G'_{\phi, K, C}, \alpha' \rangle$, then ϕ is* E-*realizable.* $\qquad\qquad\square$

This property indicates that testing whether ϕ is E-realizable can be done *incrementally* by solving the family of safety games $\langle G'_{\phi, K, C}, \alpha' \rangle$ with increasing values of $K, C \geq 0$ until either Player 1 has a winning strategy in $\langle G'_{\phi, K, C}, \alpha' \rangle$ for some K, C such that $0 \leq K \leq \mathbb{K}$, $0 \leq C \leq \mathbb{C}$, or Player 1 has no winning strategy in $\langle G'_{\phi, \mathbb{K}, \mathbb{C}}, \alpha' \rangle$.

5.2 Antichains

The LTL$_E$ realizability problem can be reduced to a family of safety games $\langle G'_{\phi, K, C}, \alpha' \rangle$ with $0 \leq K \leq \mathbb{K}$, $0 \leq C \leq \mathbb{C}$. We here show how to make more efficient the fixpoint algorithm to check whether Player 1 has a winning strategy in $\langle G'_{\phi, K, C}, \alpha' \rangle$, by avoiding to explicitly construct this safety game. This is possible because the states of $G'_{\phi, K, C}$ can be partially ordered, and the sets manipulated by the fixpoint algorithm can be compactly represented by the antichain of their maximal elements.

Partial order and antichains – Consider the safety game $\langle G'_{\phi, K, C}, \alpha' \rangle$ with $G'_{\phi, K, C} = (S = S_1 \uplus S_2, s_0, E)$ as defined above. We define the relation $\preceq \subseteq S \times S$ by

$$(F', \sigma, c') \preceq (F, \sigma, c) \text{ iff } (i) \; F' \leq F \text{ and}$$
$$(ii) \; c' \geq c$$

where $F, F' \in \mathcal{F}$, $\sigma \in \Sigma_P$, $c, c' \in \mathcal{C}$, and $F' \leq F$ iff $F'(q) \leq F(q)$ for all q. It is clear that $\preceq$ is a partial order. Intuitively, if Player 1 can win the safety game from (F, σ, c), then he can also win from all $(F', \sigma, c') \preceq (F, \sigma, c)$ as (i) it is more difficult to avoid $\top$ from F than from F', and (ii) the energy level is higher with c' than with c. Formally, $\preceq$ is a game simulation relation in the terminology of [2]. The next lemma will be useful later.

Lemma 21   – *For all $F, F' \in \mathcal{F}$ such that $F' \leq F$ and $o \cup i \in \Sigma_P$, we have $\Delta(F', o \cup i) \leq \Delta(F, o \cup i)$.*
     – *For all $c, c' \in \mathcal{C}$ such that $c' \geq c$ and $k \in \mathbb{Z}$, we have $c' \oplus k \geq c \oplus k$.*

A set $L \subseteq S$ is *closed for* $\preceq$ if $\forall (F, \sigma, c) \in S \cdot \forall (F', \sigma, c') \preceq (F, \sigma, c) \cdot (F', \sigma, c') \in L$. Let L_1 and L_2 be two closed sets, then $L_1 \cap L_2$ and $L_1 \cup L_2$ are closed. The *closure* of a set $L \subseteq S$, denoted by $\downarrow L$, is the set $\downarrow L = \{(F', \sigma, c') \in S \mid \exists (F, \sigma, c) \in L \cdot (F', \sigma, c') \preceq (F, \sigma, c)\}$. Note that if L is closed, then $\downarrow L = L$. A set $L \subseteq S$ is an *antichain* if all elements of L are incomparable for $\preceq$. Let $L \subseteq S$, we denote by $\lceil L \rceil$ the antichain composed of the maximal elements of L. If L is closed then $\downarrow \lceil L \rceil = L$, i.e. antichains are compact *canonical representations* for closed sets. The next proposition indicates how to compute antichains with respect to the union and intersection operations [18].

Proposition 22 *Let $L_1, L_2 \subseteq S$ be two antichains. Then:*

- $\downarrow L_1 \cup \downarrow L_2 = \downarrow \lceil L_1 \cup L_2 \rceil$
- $\downarrow L_1 \cap \downarrow L_2 = \downarrow \lceil L_1 \sqcap L_2 \rceil$
  where $\lceil L_1 \sqcap L_2 \rceil = \{(F_1, \sigma, c_1) \sqcap (F_2, \sigma, c_2) \mid (F_1, \sigma, c_1) \in L_1, (F_2, \sigma, c_2) \in L_2\}$, and $(F_1, \sigma, c_1) \sqcap (F_2, \sigma, c_2) : (q \mapsto \min(F_1(q), F_2(q)), \sigma, \max(c_1, c_2))$.

Fixpoint algorithm with antichains – We recall the fixpoint algorithm to check whether Player 1 has a winning strategy in the safety game $\langle G'_{\phi,K,C}, \alpha'\rangle$. This algorithm computes the fixpoint $\mathsf{Win}_1(\alpha')$ of the sequence $W_0 = \alpha', W_{k+1} = W_k \cap \{\{s \in S_1 \mid \exists (s,s') \in E \cdot s' \in W_k\} \cup \{s \in S_2 \mid \forall (s,s') \in E \cdot s' \in W_k\}\}$ for all $k \geq 0$. Player 1 has a winning strategy iff $s_0 \in \mathsf{Win}_1(\alpha')$. Let us show how to efficiently implement this algorithm with antichains.

Given $L \subseteq S$, let us denote by $\mathsf{CPre}_1(L)$ the set $\{s \in S_1 \mid \exists (s,s') \in E \cdot s' \in L\}$ and by $\mathsf{CPre}_2(L)$ the set $\{s \in S_2 \mid \forall (s,s') \in E \cdot s' \in L\}$. We have the next lemma.

Lemma 23 *If $L \subseteq S$ is a closed set, then $\mathsf{CPre}_1(L)$ and $\mathsf{CPre}_2(L)$ are also closed.*

Proof. To get the required property, it is enough to prove that if $(s,s') \in E$ and $r \preceq s$, then there exists $(r,r') \in E$ with $r' \preceq s'$.

Let us first suppose that $(s,s') \in S_1 \times S_2$. Thus, by definition of $G'_{\phi,K,C}$, we have $s = (F,j,c)$ and $s' = (F,o,c \oplus w(o))$ for some $F \in \mathcal{F}$, $c \in \mathcal{C}$, $j \in \Sigma_I$ and $o \in \Sigma_O$. Let $r = (G,j,d) \preceq (F,j,c)$, i.e. $G \leq F$ and $d \geq c$. We define $r' = (G,o,d \oplus w(o))$. Then $(r,r') \in E$, $G \leq F$ and $d \oplus w(o) \geq c \oplus w(o)$ by Lemma 21. It follows that $r' \preceq s'$.

Let us now suppose that $(s,s') \in S_2 \times S_1$. We now have $s = (F,o,c)$ and $s' = (\Delta(F,o\cup i),i,c\oplus w(i))$. Let $r = (G,o,d) \preceq (F,o,c)$, and let us define $r' = (\Delta(G,o\cup i),i,d \oplus w(i))$. By Lemma 21, we get $\Delta(G,o\cup i) \leq \Delta(F,o\cup i)$ and $d \oplus w(i) \geq c \oplus w(i)$. Therefore $r' \preceq s'$. $\qquad\square$

Notice that in the safety game $\langle G'_{\phi,K,C}, \alpha'\rangle$, the set α' is closed by definition. Therefore, by the previous lemma and Proposition 22, the sets W_k computed by the fixpoint algorithm are closed for all $k \geq 0$, and can thus be compactly represented by their antichain $\lceil W_k \rceil$. Let us show how to manipulate those sets efficiently. For this purpose, let us consider in more details the following sets of predecessors for each $o \in \Sigma_O$ and $i \in \Sigma_I$:

$$\mathsf{Pre}_o(L) = \{s \in S_1 \mid (s,s') \in E \text{ and } s' = (F,o,c) \in L, \text{ for some } F \in \mathcal{F}, c \in \mathcal{C}\}$$
$$\mathsf{Pre}_i(L) = \{s \in S_2 \mid (s,s') \in E \text{ and } s' = (F,i,c) \in L, \text{ for some } F \in \mathcal{F}, c \in \mathcal{C}\}$$

Notice that $\mathsf{CPre}_1(L) = \cup_{o \in \Sigma_O}\mathsf{Pre}_o(L)$ and $\mathsf{CPre}_2(L) = \cap_{i \in \Sigma_I}\mathsf{Pre}_i(L)$. Given $(F,o,c) \in S_2$ and $(F,i,c) \in S_1$, we define

$$\Omega(F,o,c) = \begin{cases} \{(F,i,c') \mid i \in \Sigma_I, c' = \min\{d \in \mathcal{C} \mid d \oplus w(o) \geq c\}\} & \text{if } c' \text{ exists} \\ \emptyset & \text{otherwise.} \end{cases}$$

$$\Omega(F,i,c) = \begin{cases} \{(F',o,c') \mid o \in \Sigma_O, F' = \max\{G \in \mathcal{F} \mid \Delta(G,o \cup i) \leq F\}, \\ \qquad\qquad c' = \min\{d \in \mathcal{C} \mid d \oplus w(i) \geq c\}\} & \text{if } c' \text{ exists} \\ \emptyset & \text{otherwise.} \end{cases}$$

When defining the set $\Omega(F,\sigma,c)$, we focus on the worse predecessors with respect to the partial order $\preceq$. In this definition, c' may not exist since the set $\{d \in \mathcal{C} \mid d \oplus w(\sigma) \geq c\}$ can be empty. However the set $\{G \in \mathcal{F} \mid \Delta(G,o \cup i) \leq F\}$ always contains $G : q \mapsto -1$. Moreover, even if $\preceq$ is a partial order, $\max\{G \in \mathcal{F} \mid \Delta(G,o \cup i) \leq F\}$ is unique. Indeed if $\Delta(G_1,o \cup i) \leq F$ and $\Delta(G_2,o \cup i) \leq F$, then $\Delta(G,o \cup i) \leq F$ with $G : q \mapsto \max(G_1(q),G_2(q))$.

Proposition 24 *For all $F \in \mathcal{F}$, $\sigma \in \Sigma_P$ and $c \in \mathcal{C}$, $\mathsf{Pre}_\sigma(\downarrow(F,\sigma,c)) = \downarrow\Omega(F,\sigma,c)$.*

Proof. We only give the proof for $\sigma = i \in \Sigma_I$ since the case $\sigma \in \Sigma_O$ is a particular case. We prove the two following inclusions.

1) $\mathsf{Pre}_i(\downarrow(F,i,c)) \subseteq \downarrow\Omega(F,i,c)$

Let $s' = (G,i,d) \preceq (F,i,c)$ and $s = (G',o,d')$ such that $(s,s') \in E$. We have to show that $s \preceq \Omega(F,i,c)$. As $(s,s') \in E$, we have $\Delta(G',o \cup i) = G \leq F$ and $d' \oplus w(i) = d \geq c$. It follows that $(G',o,d') \preceq \Omega(F,i,c)$ by definition of $\Omega(F,i,c)$.

2) $\downarrow\Omega(F,i,c) \subseteq \mathsf{Pre}_i(\downarrow(F,i,c))$

Let $(F',o,c') \in \Omega(F,i,c)$ and $s = (G',o,d') \preceq (F',o,c')$. We have to show that there exists $(s,s') \in E$ with $s' \preceq (F,i,c)$. By definition of $\Omega(F,i,c)$, we have $\Delta(F',o\cup i) \leq F$ and $c' \oplus w(i) \geq c$. As $G' \leq F'$ and $d' \geq c'$, it follows that $\Delta(G',o\cup i) \leq \Delta(F',o\cup i) \leq F$ and $d' \oplus w(i) \geq c' \oplus w(i) \geq c$ by Lemma 21. Therefore with $s' = (\Delta(G',o\cup i),i,d'\oplus w(i))$, we have $(s,s') \in E$ and $s' \preceq (F,i,c)$. Thus $(G',o,d') \in \mathsf{Pre}_i(\downarrow(F,i,c))$.

$\square$

Propositions 22 and 24 indicate how to limit to antichains the computation steps of the fixpoint algorithm.

Corollary 25 *If $L \subseteq S$ is an antichain, then* $\mathsf{CPre}_1(L) = \bigcup_{o\in\Sigma_O} \bigcup_{(F,o,c)\in L} \downarrow\Omega(F,o,c)$ *and* $\mathsf{CPre}_2(L) = \bigcap_{i\in\Sigma_I} \bigcup_{(F,i,c)\in L} \downarrow\Omega(F,i,c)$.

Optimizations – The definition of $\Omega(F,i,c)$ requires to compute $\max\{G \in \mathcal{F} \mid \Delta(G,o\cup i) \leq F\}$. This computation can be done more efficiently using the operator $\ominus : \mathcal{K} \times \{0,1\} \to \mathcal{K}$ defined as follows: $k \ominus b = \top$ if $k = \top$, $k \ominus b = -1$ if $k \neq \top$, $k - b \leq -1$, and $k \ominus b = k - b$ in all other cases. Indeed, using Lemma 4 of [18], one can see that

$$\max\{G \in \mathcal{F} \mid \Delta(G,o\cup i) \leq F\} = q \mapsto \min\{F(q') \ominus (q' \in \alpha') \mid q' \in \delta(q,o\cup i)\}.$$

It is possible to reduce the size of the safety game $\langle G'_{\phi,K,C}, \alpha'\rangle$ such that $S_1 = \{(F,c) \mid F \in \mathcal{F}, c \in \mathcal{C}\}$ instead of $\{(F,i,c) \mid F \in \mathcal{F}, i \in \Sigma_I, c \in \mathcal{C}\}$. We refer to the proof of Proposition 8 and footnote 11 for the justification.

$\mathsf{LTL_E}$*-synthesis –* If Player 1 has a winning strategy in the safety game $\langle G'_{\phi,K,C}, \alpha'\rangle$, that is, the given formula ϕ is E-realizable, then it is easy to contruct a Moore machine $\mathcal{M}$ that realizes it. As described in [18], $\mathcal{M}$ can be constructed from the antichain $\lceil\mathsf{Win}_1(\alpha')\rceil$ computed by the fixpoint algorithm, with the advantage of having a small size bounded by the size of $\lceil\mathsf{Win}_1(\alpha')\rceil$ (see Section 7.2).

Forward algorithm – The proposed fixpoint algorithm works in a backward manner. In [18], the authors propose a variant of the OTFUR algorithm of [10] that computes a winning strategy for Player 1 (if it exists) in a forward fashion, starting from the initial state of the safety game. This forward algorithm can be adapted to the safety game $\langle G'_{\phi,K,C}, \alpha'\rangle$. As for the backward fixpoint algorithm, it is not necessary to construct the game explicitly and antichains can again be used [18]. Compared to the backward algorithm, the forward algorithm has the following advantage: it only computes winning states (F,σ,c) (for Player 1) which are reachable from the initial state. Nevertheless, it computes a single winning strategy if it exists, whereas the backward algorithm computes a fixpoint from which we can easily enumerate the set of all winning strategies (in the safety game).

6 Extension to multi-dimensional weights

Multi-dimensional $\mathsf{LTL_{MP}}$ and $\mathsf{LTL_E}$ realizability problems – The $\mathsf{LTL_{MP}}$ and $\mathsf{LTL_E}$ realizability problems can be naturally extended to multi-dimensional weights. Given P, we define a weight function $w : \mathsf{Lit}(P) \to \mathbb{Z}^m$, for some dimension $m \geq 1$. The concepts of energy level EL, mean-payoff value MP, and value Val are defined similarly. Given an LTL formula ϕ over $\langle P,w\rangle$ and a threshold $\nu \in \mathbb{Q}^m$, the *multi-dimensional $\mathsf{LTL_{MP}}$ realizability problem (under finite memory)* asks to decide whether there exists a (finite-memory) strategy λ_O of Player O such that $\mathsf{Val}(\mathsf{Outcome}(\lambda_O,\lambda_I)) \geq \nu$ against all strategies λ_I of Player I.[12] The *multi-dimensional $\mathsf{LTL_E}$ realizability problem* asks to decide whether there exists a strategy λ_O of Player O and an initial credit $c_0 \in \mathbb{N}^m$ such that for all strategies λ_I of Player I, *(i)* $u = \mathsf{Outcome}(\lambda_O,\lambda_I) \models \phi$, *(ii)* $\forall n \geq 0,\ c_0 + \mathsf{EL}(u(n)) \geq (0,\dots,0)$.

---

[12] With $a \geq b$, we mean $a_i \geq b_i$ for all i, $1 \leq i \leq m$.

Computational complexity – The 2ExpTime-completeness of the $\mathsf{LTL_{MP}}$ and $\mathsf{LTL_E}$ realizability problems have been stated in Theorem 2 and 11 in one dimension. In the multi-dimensional case, we have the next result.

Theorem 26. *The multi-dimensional $\mathsf{LTL_{MP}}$ realizability problem under finite memory and the multi-dimensional $\mathsf{LTL_E}$ realizability problem are in co-N2ExpTime.*

Before giving the proof, we need to introduce *multi-mean-payoff games* and *multi-energy games* and some related results. Those games are defined as in Section 3.2 with the only difference that the weight function w assigns an m-tuple of weights to each edge of the underlying graph. The next proposition extends Proposition 13 to multiple dimensions.

Proposition 27 ([13]) *Let $\langle G, w, m \rangle$ be a game with $w : E \to \mathbb{Z}^m$. Let $\nu \in \mathbb{Q}^m$ be a threshold and λ_1 be a finite-memory strategy for Player 1. Then λ_1 is winning in the multi-mean-payoff parity game $\langle G, w, p \rangle$ with threshold ν iff λ_1 is winning in the multi-energy parity game $\langle G, w - \nu \rangle$ for some initial credit $c_0 \in \mathbb{N}^m$.*

In [15], the authors study the initial credit problem for multi-energy parity games $\langle G, w, p, m \rangle$. They introduce the notion of self-covering tree[13] associated with the game, and show that its depth is bounded by a constant $l = 2^{(h-1)\cdot|S|} \cdot (W \cdot |S| + 1)^{c \cdot m^2}$ where $|S|$ is the number of states of G, W is its largest absolute weight, h is the highest number of outgoing edges on any state of S, m is the dimension, and c is a constant independent of the game. The next proposition states that multi-energy parity games reduce to multi-energy games.

Proposition 28 ([15]) *Let $\langle G, w, p, m \rangle$ be a multi-energy parity game with a priority function $p : S \to \{0, 1, \ldots, 2 \cdot d\}$ and a self-covering tree of depth bounded by l. Then one can construct a multi-energy game $\langle G, w', m' \rangle$ with $m' = m + d$ dimensions and a largest absolute weight W' bounded by l, such that a strategy is winning for Player 1 in $\langle G, w, p, m \rangle$ iff it is winning in $\langle G, w', m' \rangle$.*

The next results are taken from [13] and [15].

Theorem 29 ([13,15]).

- *The initial credit problem for a multi-energy game is coNP-Complete.*
- *If Player 1 has a winning strategy for the initial credit problem in a multi-energy parity game, then he can win with a finite-memory strategy of at most exponential size.*
- *Let $\langle G, w, m \rangle$ be a multi-energy game with a self-covering tree of depth bounded by l. If Player 1 has a winning strategy for the initial credit problem, then he can win with an initial credit $(C, \ldots, C) \in \mathbb{N}^m$ such that $C = 2 \cdot l \cdot W$.[14]*

Proof (of Theorem 26). We proceed as in the proof of the theorems 2 and 11 by reducing the $\mathsf{LTL_{MP}}$ (resp. $\mathsf{LTL_E}$) realizability of formula ϕ to a multi-mean-payoff (resp. multi-energy) parity game $\langle G_\phi, w, p, m \rangle$. By Proposition 27, it is enough to study the multi-dimensional $\mathsf{LTL_E}$ realizability problem. We reduce the multi-energy parity game $\langle G_\phi, w, p, m \rangle$ to a multi-energy game $\langle G_\phi, w', m' \rangle$ as described in Proposition 28. Careful computations show that the multi-dimensional $\mathsf{LTL_E}$ realizability problem is in co-N2ExpTime, by using Theorems 3 and 29. □

Theorem 26 states the complexity of the $\mathsf{LTL_{MP}}$ realizability problem under finite memory. Notice that it is reasonable to ask for finite-memory (instead of any) winning strategies. Indeed, the previous proof indicates a reduction to a multi-mean-payoff game; winning strategies for Player 1 in such games require infinite memory in general; however, if Player 1 has a winning strategy for threshold ν, then he has a finite-memory one for threshold $\nu - \epsilon$ for all $\epsilon > 0$ [25].

---

[13] See [15] for the definition and results.

[14] This result is extended in [15] to multi-energy parity games thanks to Proposition 28.

Safraless algorithm – As done in one dimension, we can similarly show that the multi-dimensional $\mathsf{LTL_E}$-realizability problem can be reduced to a safety game for which there exist symbolic antichain-based algorithms. The multi-dimensional $\mathsf{LTL_{MP}}$-realizability problem under finite memory can be solved similarly thanks to Proposition 27.

Theorem 30. *Let ϕ be an LTL formula. Then one can construct a safety game in which Player 1 has a winning strategy iff ϕ is E-realizable.*

Proof. The proof is very similar to the one of Theorem 19. We only indicate the differences.

First, let $\langle G_\phi, w, p\rangle$ be the multi-energy parity game associated with ϕ as described in the proof of Theorem 26. By Theorem 29 and Proposition 12 adapted to this multi-dimensional game, we know that if ϕ is E-realizable, then Player O has a finite-memory strategy with a memory size M that is at most exponential in the size of the game.

Second, we need to work with multi-energy automata over a weighted alphabet $\langle \Sigma_P, w\rangle$, such that w is a function over $\mathsf{Lit(P)}$ that assigns m-tuples of weights instead of a single weight.

Third, from a UCB $\mathcal{A}$ with n states such that $L_{\mathrm{ucb}}(\mathcal{A}) = [\![\phi]\!]$, we construct, similarly as in the one-dimensional case, a safety game $\langle G'_{\phi,\mathbb{K},(\mathbb{C},...,\mathbb{C})}, \alpha'\rangle$ whose positions store a counter for each state of $\mathcal{A}$ and an energy level for each dimension. The constants $\mathbb{K}$ and $\mathbb{C}$ are defined differently from the one-dimensional case: $\mathbb{K} = n \cdot M$, and $\mathbb{C}$ is equal to the constant C of Theorem 29. □

Antichain-based algorithms – Similarly to the one-dimensional case, testing whether an LTL formula ϕ is E-realizable can be done incrementally by solving a family of safety games related to the safety game given in Theorem 30. These games can be symbolically solved by the antichain-based backward and forward algorithms described in Section 5.

7 Experiments

In the previous sections, in one or several dimensions, we have shown how to reduce the $\mathsf{LTL_{MP}}$ under finite memory and $\mathsf{LTL_E}$ realizability problems to a safety games, and how to derive symbolic antichain-based algorithms. This approach has been implemented in our tool $\mathsf{Acacia+}$. We briefly present this tool and give some experimental results.

7.1 Tool Acacia+

In [4], we present $\mathsf{Acacia+}$, a tool for LTL synthesis using antichain-based algorithms. The main advantage of this tool, in comparison with other LTL synthesis tools, is to generate compact strategies that are easily usable in practice. This aspect can be very useful in many application scenarios like synthesis of control code from high-level LTL specifications, debugging of unrealizable LTL specifications by inspecting compact counter strategies, and generation of small deterministic Büchi or parity automata from LTL formulas (when they exist) [4].

$\mathsf{Acacia+}$ is now extended to the synthesis from LTL specifications with mean-payoff objectives in the multi-dimensional setting. As explained in the previous sections, it solves incrementally a family of safety games, depending on some values K and C, to test whether a given specification ϕ is MP-realizable under finite memory. The tool takes as input an LTL formula ϕ with a partition of its set P of atomic signals, a weight function $w : \mathsf{Lit}(P) \mapsto \mathbb{Z}^m$, a threshold value $\nu \in \mathbb{Q}^m$, and two bounds $K \in \mathbb{Z}$ and $C \in \mathbb{Z}^m$ (the user can specify additional parameters to define the incremental policy). It then searches for a finite-memory winning strategy for Player O, within the bounds of K and C, and outputs a Moore machine if such a strategy exists. The last version of $\mathsf{Acacia+}$ can be downloaded at `http://lit2.ulb.ac.be/acaciaplus/` and it can also be used directly online via a web interface. Moreover, many benchmarks and results tables are available on the website.

7.2 Experiments

In this section, we present some experiments. They have been done on a Linux platform with a 3.2GHz CPU (Intel Core i7) and 12GB of memory.

Approaching the optimal value – Let us come back to Example 2, where we have given a specification ϕ together with a 1-dimensional mean-payoff objective. For the optimal value ν_ϕ, we have shown that no finite-memory strategy exists, but finite-memory ϵ-optimal strategies exist for all $\epsilon > 0$. In Table 1, we present the experiments done for some values of $\nu_\phi - \epsilon$. The output strategies for the system behave

Table 1. Acacia+ on the specification of Example 2 with increasing threshold values. The column ν gives the threshold values, K and C the minimum values required to obtain a winning strategy, $|\mathcal{M}|$ the size of the Moore machine representing the strategy and $time$ the execution time (in seconds). Note that the execution times are given for the forward algorithm applied to the safety game with values K and C (and not with smaller ones).

ν	K	C	$|\mathcal{M}|$	time (s)
-1.2	4	7	5	0.01
-1.02	49	149	50	0.05
-1.002	499	1499	500	0.34
-1.001	999	2999	1000	0.89
-1.0002	4999	14999	5000	15.49
-1.0001	9999	29999	10000	59.24
-1.00005	19999	99999	20000	373

as follows: grant the second client ($|\mathcal{M}| - 1$) times, then grant once client 1, and start over. Thus, the system almost always plays $g_2 w_1$, except every $|\mathcal{M}|$ steps where he has to play $g_1 w_2$. Obviously, these strategies are the smallest ones that ensure the corresponding threshold values. They can also be compactly represented by a two-state automaton with a counter that counts up to $|\mathcal{M}|$. With $\nu = -1.001$ of Table 1, let us emphasize the interest of using antichains in our algorithms. The underlying state space manipulated by our symbolic algorithm is huge: since $K = 999$, $C = 2999$ and the number of automata states is 8, the number of states is around 10^{27}. However the fixpoint computed backwardly is represented by an antichain of size 2004 only.

No unsollicited grants – The major drawback of the strategies presented in Table 1 is that many unsollicited grants might be sent since the server grants the resource access to the clients in a round-robin fashion (with a longer access for client 2 than for client 1) without taking care of actual requests made by the clients. It is possible to express in LTL the fact that no unsollicited grants occur, but it is cumbersome. Alternatively, the $\mathsf{LTL_{MP}}$ specification can be easily rewritten with a multi-dimensional mean-payoff objective to avoid those unsollicited grants, as shown in Example 3.

Example 3. We consider the client-server system of Examples 1 and 2 with the additional requirement that the server does not send unsollicited grants. This property can be naturally expressed by keeping the inital LTL specification ϕ and proposing a multi-dimensional mean-payoff objective as follows. A new dimension is added by client, such that a request (resp. grant) signal of client i has a reward (resp. cost) of 1 on his new dimension. More precisely, let ϕ and P as in Example 2, we define $w : \mathsf{Lit}(P) \to \mathbb{Z}^3$ as the weight function such that $w(r_1) = (0, 1, 0)$, $w(r_2) = (0, 0, 1)$, $w(g_1) = (0, -1, 0)$, $w(g_2) = (0, 0, -2)$, $w(w_1) = (-1, 0, 0)$, $w(w_2) = (-2, 0, 0)$ and $w(l) = (0, 0, 0)$, $\forall l \in \mathsf{Lit}(P) \setminus \{r_1, r_2, g_1, g_2, w_1, w_2\}$.

For threshold $\nu = (-1, 0, 0)$, there is no hope to have a finite-memory strategy (see Example 2). For threshold $\nu = (-1.2, 0, 0)$ and values $K = 4$, $C = (7, 1, 1)$, Acacia+ outputs a finite-memory strategy computed by the backward algorithm, as depicted in Figure 1. In this figure, the strategy is represented by a transition system where the red state is the initial state, and the transitions are labeled with symbols $o|i$ with $o \in \Sigma_O$ and $i \in \Sigma_I$. Notice that the labels of all outgoing transitions of a state share the same o part (since we deal with a strategy). This transition system can be seen as a Moore machine $(\mathcal{M}, m_0, \alpha_U, \alpha_N)$ with the same state space (the set M of memory states), and such that for each transition from m to m' labeled by $o \cup i$, we have $\alpha_U(m, i) = m'$ and $\alpha_N(m) = o$. We can verify that no unsollicited grant is done if the server plays according to this strategy. Moreover, this is the smallest strategy to ensure a threshold of $(-1.2, 0, 0)$ against the most demanding behavior of the clients, i.e. when they both make requests all the time (see states 3 to 7), and that avoid unsollicited grants against any other behaviors of the clients (see states 0 to 2).
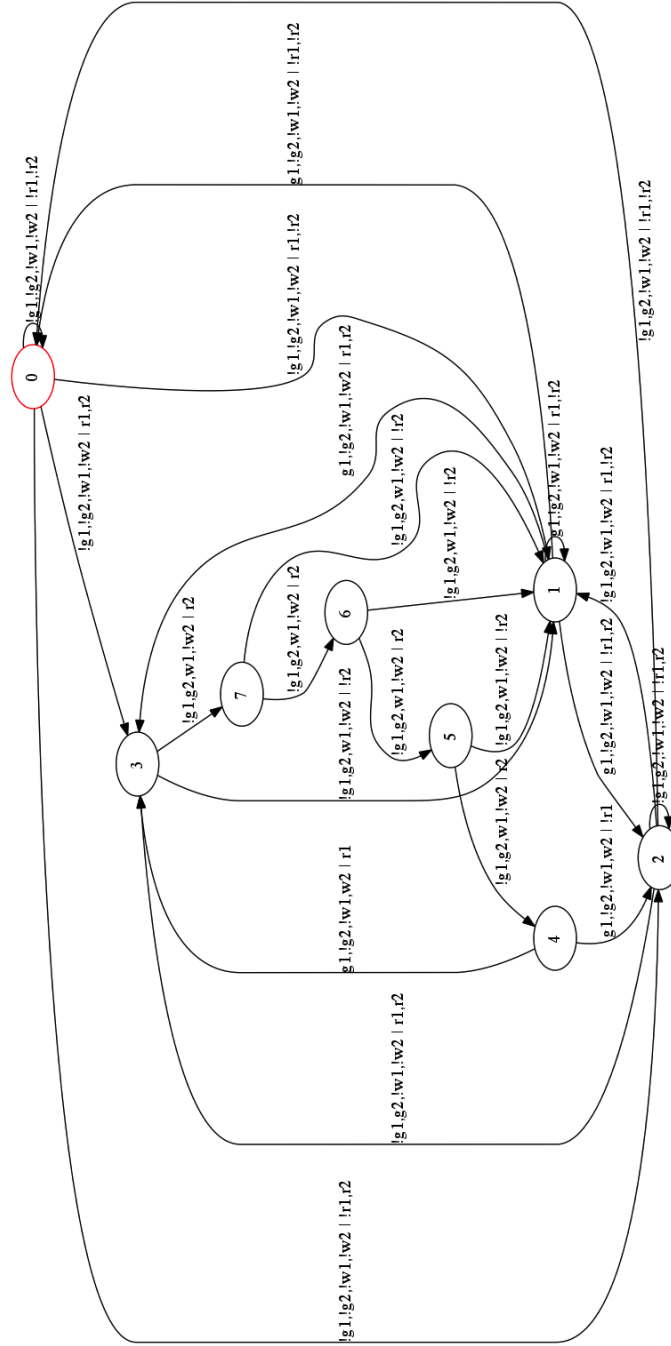
Fig. 1. Strategy output by Acacia+ for the specification of Example 3, threshold $\nu = (-1.2, 0, 0)$ and values $K = 4$, $C = (7, 1, 1)$, using the backward algorithm

From Example 3, we derive a benchmark of multi-dimensional examples parameterized by the number of clients making requests to the server. Some experimental results of Acacia+ on this benchmark are synthetized in Table 2.

Table 2. Acacia+ on the Shared Resource Arbiter benchmark parameterized by the number of clients, with the forward algorithm. The column c gives the number of clients, ν the threshold, K (resp. C) the minimum value (resp. vector) required to obtain a winning strategy, $|\mathcal{M}|$ the size of the Moore machine representing the strategy and $time$ the total execution time (in seconds).

c	ν	K	C	$\|\mathcal{M}\|$	time (s)
2	$(-1.2, 0, 0)$	4	$(7, 1, 1)$	11	0.02
3	$(-2.2, 0, 0, 0)$	9	$(19, 1, 1, 1)$	27	0.12
4	$(-3.2, 0, 0, 0, 0)$	14	$(12, 1, 1, 1, 1)$	65	1.81
5	$(-4.2, 0, 0, 0, 0, 0)$	19	$(29, 1, 1, 1, 1, 1)$	240	59
6	$(-5.2, 0, 0, 0, 0, 0, 0)$	24	$(17, 1, 1, 1, 1, 1, 1)$	1716	4207

Approching the Pareto curve – As last experiment, we consider the 2-client LTL$_{MP}$ specification of Example 3 where we split the first dimension of the weight function into two dimensions, such that $w(w_1) = (-1, 0, 0, 0)$ and $w(w_2) = (0, -2, 0, 0)$. With this new specification, since we have several dimensions, there might be several optimal values for the pairwise order, corresponding to trade-offs between the two objectives that are (i) to quickly grant client 1 and (ii) to quickly grant client 2. In this experiment, we are interested in approaching, by hand, the *Pareto curve*, which consists of all those optimal values, i.e. to find finite-memory strategies that are incomparable w.r.t. the ensured thresholds, these thresholds being as large as possible. We give some such thresholds in Table 3, along with minimum K and C and strategies size. It is difficult to automatize the construction of the Pareto curve. Indeed, Acacia+ cannot test (in reasonable time) whether a formula is MP-unrealizable for a given threshold, since it has to reach the huge theoretical bound on K and C. This raises two interesting questions that we let as future work: how to decide efficiently that a formula is MP-unrealizable for a given threshold, and how to compute points of the Pareto curve efficiently.

Table 3. Acacia+ to approach Pareto values. The column ν gives the threshold, relatively close to the Pareto curve, K (resp. C) the minimum value (resp. vector) required to obtain a winning strategy, $|\mathcal{M}|$ the size of the Moore machine representing the strategy.

ν	K	C	$\|\mathcal{M}\|$
$(-0.001, -2, 0, 0)$	999	$(1999, 1, 1, 1)$	2001
$(-0.15, -1.7, 0, 0)$	55	$(41, 55, 1, 1)$	42
$(-0.25, -1.5, 0, 0)$	3	$(7, 9, 1, 1)$	9
$(-0.5, -1, 0, 0)$	1	$(3, 3, 1, 1)$	5
$(-0.75, -0.5, 0, 0)$	3	$(9, 7, 1, 1)$	9
$(-0.85, -0.3, 0, 0)$	42	$(55, 41, 1, 1)$	9
$(-1, -0.01, 0, 0)$	199	$(1, 399, 1, 1)$	401

References

1. C. S. Althoff, W. Thomas, and N. Wallmeier. Observations on determinization of Büchi automata. *Theor. Comput. Sci.*, 363(2):224–233, 2006.
2. R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In D. Sangiorgi and R. de Simone, editors, *CONCUR*, volume 1466 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 1998.

3. R. Bloem, K. Chatterjee, T. A. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In Bouajjani and Maler [6], pages 140–156.
4. A. Bohy, V. Bruyère, E. Filiot, N. Jin, and J.-F. Raskin. Acacia+, a tool for LTL synthesis. In P. Madhusudan and S. A. Seshia, editors, *CAV*, volume 7358 of *Lecture Notes in Computer Science*, pages 652–657. Springer, 2012.
5. U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. In *LICS*, pages 43–52. IEEE Computer Society, 2011.
6. A. Bouajjani and O. Maler, editors. *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*. Springer, 2009.
7. P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, and J. Srba. Infinite runs in weighted timed automata with energy constraints. In F. Cassez and C. Jard, editors, *FORMATS*, volume 5215 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2008.
8. P. Bouyer, N. Markey, J. Olschewski, and M. Ummels. Measuring permissiveness in parity games: Mean-payoff parity games revisited. In T. Bultan and P.-A. Hsiung, editors, *ATVA*, volume 6996 of *Lecture Notes in Computer Science*, pages 135–149. Springer, 2011.
9. L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J.-F. Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011.
10. F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In M. Abadi and L. de Alfaro, editors, *CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2005.
11. A. Chakrabarti, L. de Alfaro, T. A. Henzinger, and M. Stoelinga. Resource interfaces. In R. Alur and I. Lee, editors, *EMSOFT*, volume 2855 of *Lecture Notes in Computer Science*, pages 117–133. Springer, 2003.
12. K. Chatterjee and L. Doyen. Energy parity games. In S. Abramsky, C. Gavoille, C. Kirchner, F. Meyer auf der Heide, and P. G. Spirakis, editors, *ICALP (2)*, volume 6199 of *Lecture Notes in Computer Science*, pages 599–610. Springer, 2010.
13. K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Generalized mean-payoff and energy games. In K. Lodaya and M. Mahajan, editors, *FSTTCS*, volume 8 of *LIPIcs*, pages 505–516. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
14. K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Mean-payoff parity games. In *LICS*, pages 178–187. IEEE Computer Society, 2005.
15. K. Chatterjee, M. Randour, and J.-F. Raskin. Strategy synthesis for multi-dimensional quantitative objectives. In M. Koutny and I. Ulidowski, editors, *CONCUR*, volume 7454 of *Lecture Notes in Computer Science*, pages 115–131. Springer, 2012.
16. R. Ehlers. Symbolic bounded synthesis. In T. Touili, B. Cook, and P. Jackson, editors, *CAV*, volume 6174 of *Lecture Notes in Computer Science*, pages 365–379. Springer, 2010.
17. E. Filiot, N. Jin, and J.-F. Raskin. An antichain algorithm for ltl realizability. In Bouajjani and Maler [6], pages 263–277.
18. E. Filiot, N. Jin, and J.-F. Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011.
19. E. Grädel, W. Thomas, and T. Wilke. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.
20. T. A. Henzinger. Quantitative reactive models. In R. B. France, J. Kazmeier, R. Breu, and C. Atkinson, editors, *MoDELS*, volume 7590 of *Lecture Notes in Computer Science*, pages 1–2. Springer, 2012.
21. B. Jobstmann and R. Bloem. Optimizations for LTL synthesis. In *Proceedings of the 6th International Conference on Formal Methods in Computer Aided Design (FMCAD)*, pages 117–124. IEEE Computer Society, 2006.
22. O. Kupferman and M. Y. Vardi. Safraless decision procedures. In *FOCS*, pages 531–542. IEEE Computer Society, 2005.
23. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989.
24. S. Schewe and B. Finkbeiner. Bounded synthesis. In K. S. Namjoshi, T. Yoneda, T. Higashino, and Y. Okamura, editors, *ATVA*, volume 4762 of *Lecture Notes in Computer Science*, pages 474–488. Springer, 2007.
25. Y. Velner, K. Chatterjee, L. Doyen, T. A. Henzinger, A. Rabinovich, and J.-F. Raskin. The complexity of multi-mean-payoff and multi-energy games. *CoRR*, abs/1209.3234, 2012.
26. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996.