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 integer 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\textsubscript{MP} synthesis and the LTL\textsubscript{MP} realizability problem asks to check whether such a system exists. By reduction to two-player mean-payoff parity games, we first show that LTL\textsubscript{MP} realizability is not more difficult than LTL realizability: it is 2ExpTime-Complete. While infinite memory strategies are required to realize LTL\textsubscript{MP} specifications in general, we show that $\epsilon$-optimality can be obtained with finite-memory strategies, for any $\epsilon > 0$. To obtain efficient algorithms 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 one to express Boolean properties 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

* Author supported by ERC Starting Grant (279499: inVEST).

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 [15] (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 integer 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 [22], 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 $-c_0$ [5].

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 synthesize a server that eventually grants any client request, and that only grants one request at a time. This can be formalized in LTL as the conjunction of the three following formulas, 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:

$$
\phi_1 = \Box(r_1 \rightarrow X(w_1 U g_1)) \quad \phi_2 = \Box(r_2 \rightarrow X(w_2 U g_2)) \quad \phi_3 = \Box(\neg g_1 \lor \neg g_2)
$$

Intuitively, $\phi_1$ (resp. $\phi_2$) 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 $\phi_3$ stands for mutual exclusion. Let $\phi = \phi_1 \land \phi_2 \land \phi_3$.

The formula $\phi$ 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 $\phi$ 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 unsolicited 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 other examples in the paper and describe the solutions that we obtain automatically with our algorithms.