Model Checking of Optimal LTL and ASAP Properties

Davide Bresolin[1], Filippo Fantinato[1,2] and Stefano Tonetta[2]

[1]Università di Padova, Padova, Italy
[2]Fondazione Bruno Kessler, Trento, Italy

Abstract

Model checking has long been employed as a method for the formal verification of control systems, with a focus on ensuring correctness and safety. However, in practical scenarios (e.g., robotics, aviation and aerospace), simply verifying whether a control system satisfies a given property may not suffice. There is often the requisite to optimize system behavior with respect to certain criteria, such as response time. For instance, in verifying a reachability property, one may be interested in knowing if the controller reaches the goal as soon as possible (ASAP). Despite the simplicity of such requirement, its formalization has not yet been addressed in the literature and requires to reason about the strategy of the controller and the cost of the executions in closed-loop with the given environment. More in general, this paper proposes the formalization and verification of properties for controllers that must satisfy a temporal logic specification optimally, i.e., in the best way possible given the behavior on the plant to be controlled. This relies on and is parametrized by a quantitative semantics for temporal logic. We focus on linear-time temporal logic (LTL), for which various quantitative semantics have been defined. In order to characterize the fulfillment of LTL properties as soon as possible (ASAP), we introduce a new quantitative semantics related to the length of the shortest informative prefix. Finally, we focus on ASAP co-safety properties and reduce the optimal model checking to standard qualitative reactive synthesis. We provide a proof of concept demonstration of the reduction with nuXmv.

1. Introduction

Model checking has long been employed as a method for the formal verification of control systems, with a focus on ensuring correctness and safety. However, in practical scenarios (e.g., robotics, aviation and aerospace), simply verifying whether a control system satisfies a given property may not suffice. There is often a desire to optimize system behavior with respect to certain criteria, such as response time. For instance, in verifying a reachability property, one may be interested in knowing if the controller reaches the goal as soon as possible (ASAP). Despite the simplicity of such requirement, its formalization has not yet been addressed in the literature and requires to reason about the strategy of the controller and the cost of the executions in closed-loop with the given environment. More in general, this paper formalizes the problem of verifying that a control system not only satisfies a specified LTL property, but also it does it in the best way possible given the assumption on the environment. The problem is parametric with respect to a quantitative semantics that defines the value the controller should optimize. As second contribution, we propose a novel quantitative semantics for LTL tailored to capture the ASAP requirement for temporal reachability properties and, as third contribution, we focus on a co-safety fragment of LTL, namely negation of formulas expressed in LTL_{EBR} [1], and reduce the optimal model checking problem with the ASAP semantics to a qualitative realizability problem. Finally, to demonstrate the efficacy of our approach, we have implemented our methodology and applied it to various examples using nuXmv [2], a state-of-the-art model checker. We tested the optimal model checking procedure on various co-safety properties and on some scalable benchmarks.

The rest of the paper is organized as follows: Section 2 describes a motivating example; Section 3 recalls the background definitions and results; Section 4 defines the optimal model checking problem; Section 5 introduces the new ASAP semantics; Section 6 reduces the ASAP optimal model checking problem to reactive synthesis; in Section 7, we compare with related work; in Section 8, we report on the experimental results, and finally in Section 9, we draw some conclusions.

OVERLAY 2024, 6th International Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis
davide.bresolin@unipd.it (D. Bresolin); filippo.fantinato.2@studenti.unipd.it (F. Fantinato); tonettas@fbk.eu (S. Tonetta)

2. Example

We introduce the optimal model checking problem with a simple illustrative example of an agent that must reach a goal position avoiding an opponent, for example a fox that must reach a chicken and it is hindered by a dog, as shown in Figure 1.

The fox and the dog move in a grid, while the chicken stays in the upper right corner. The possible movements are: move up, down, left, right, or stop. Suppose that the fox moves faster than the dog (i.e. the fox can move at each step while the dog moves every other step) and that the dog is not standing still. The strategy of the fox can be to go towards the chicken unless the dog is on the way and in that case, it just stops until the dog moves and then it goes ahead. However, such a strategy does not reach the goal as soon as possible. Since the fox is quicker than the dog, a better strategy is to pass around the dog.

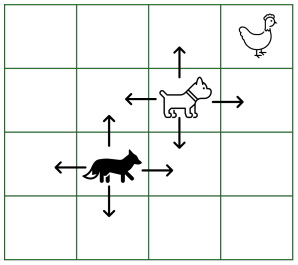

Figure 1: The fox-dog-chicken example.

The controller D and the plant P are specified as transition systems. The property is specified in LTL as $\neg \mathit{bad\_states}\, \mathbf{U}\, \mathit{goal}$ where *bad_states* are the states where the fox and the dog are in the same position, while *goal* is the state where the fox and the chicken are in the same cell of the grid. We formalize and solve the problem of checking whether D satisfies the property ASAP with the given plant P.

3. Notations and Preliminary Definitions

In the following, we consider Symbolic Transition Systems (STSs), also known as synchronous state machines, to represent the behavior of reactive systems, i.e. systems whose role is to maintain an ongoing interaction with their environment.

Given a set of propositional variables Σ, an STS S over Σ is a tuple $\langle V, \Sigma, I, T \rangle$, where V is a set of state variables disjoint from Σ, I is a formula over V, and T is a formula over $V \cup \Sigma \cup V'$. We assume the reader to be familiar with the standard notions of paths, traces, and products of STSs.

In order to specify properties, we consider Linear Temporal Logic with future and past operators, denoted by LTL and interpreted over infinite sequences of states. Two notable classes of properties that can be expressed in LTL are safety and co-safety properties. Safety properties are those properties which can be refuted by a bad-prefix, i.e., a finite computation than cannot be extended to an infinite computation that satisfies the formula. Conversely, co-safety properties are those properties which can be verified by identifying a good-prefix, i.e., a finite computation such that all infinite extensions satisfy the formula. In this paper we use the syntactic fragment LTL_{EBR} [1] to express safety properties, and the dual fragment co-LTL_{EBR} for co-safety ones.

Definition 3.1 (The logic co-LTL_{EBR})**.** *Let $a, b \in \mathbb{N}$, a LTL_{EBR} formula χ is inductively defined as follows:*

$$\eta := p \mid \neg\eta \mid \eta_1 \vee \eta_2 \mid \mathbf{Y}\eta \mid \eta_1 \mathbf{S} \eta_2 \qquad \text{(Pure Past Layer)}$$

$$\psi := \eta \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \mathbf{X}\psi \mid \psi_1 \mathbf{U}^{[a,b]} \psi_2 \qquad \text{(Bounded Future Layer)}$$

$$\phi := \psi \mid \phi_1 \wedge \phi_2 \mid \mathbf{X}\phi \mid \mathbf{F}\phi \mid \psi\, \mathbf{U}\, \phi \qquad \text{(Future Layer)}$$

$$\chi := \phi \mid \chi_1 \vee \chi_2 \mid \chi_1 \wedge \chi_2 \qquad \text{(Boolean Layer)}$$

In the rest of the paper, we assume that $\Sigma = C \cup U$ is divided into two disjoint sets C and U of, respectively, controllable and uncontrollable variables.

Definition 3.2. *An arena for a reachability game is given by a deterministic STS $S = \langle V, \Sigma, I, T \rangle$ and a formula f over V, called winning condition. We say that a strategy for controller $g\colon (2^U)^+ \to 2^C$ wins the reachability game with arena S and winning condition f iff for all sequences of choices made by*

environment $u = u_1, u_2, \cdots \in (2^U)^\omega$*, there exists* n *such that the* n*-th state of the unique path over the trace* $u_1 \cdot g(u_1), u_2 \cdot g(u_1 u_2), \ldots$ *satisfies* f*.*

Reactive synthesis is the problem of translating a logical specification into a reactive system that is guaranteed to satisfy the specification for all possible behaviors of its environment.

Definition 3.3 (Reactive Synthesis problem)**.** *Let* ϕ *be a temporal formula over the alphabet* $\Sigma = C \cup U$*. We say that* ϕ *is realizable if and only if there exists a strategy* $g\colon (2^U)^+ \to 2^C$ *such that for all* $u = (u_1, u_2, \ldots) \in (2^U)^\omega$*, it holds that the sequence satisfies* $u_1 \cdot g(u_1), u_2 \cdot g(u_1 u_2), \ldots \models \phi$*. The synthesis problem is the decision problem to say whether* ϕ *is realizable or not. If* ϕ *is realizable, the corresponding strategy* g *is computed.*

For co-safety properties, reactive synthesis can be reduced to finding a winning strategy in a reachability game defined from the formula. In [1] is shown how reactive synthesis from LTL_{EBR} formulas can be reduced to solving a safety game over a monitor built directly from the specifications. Such monitor accepts only those traces satisfying the formula from which is built, while it goes into error state whenever a trace not satisfying the formula is met. Thanks to the duality between safety and reachability games, it is possible to synthesize a co-LTL_{EBR} formula exploiting the monitor built from the corresponding LTL_{EBR} formula, i.e. the negation.

In [3], the synthesis problem is extended to variants of LTL with quantitative semantics such as $\text{LTL}[\mathscr{F}]$ and $\text{LTL}^{\text{disc}}[\mathscr{D}]$. In general, quantitative semantics assign a real value $[\![\sigma, \phi]\!] \in [0, 1]$ to a formula ϕ over a sequence σ. As in [3], we extend the evaluation of a formula to a system as $[\![S, \phi]\!] := \inf \{[\![\sigma, \phi]\!] \mid \sigma \in \mathscr{L}(S)\}$, and thus to a strategy g as $[\![g, \phi]\!] := \inf \{[\![g(u), \phi]\!] \mid u \in (2^U)^\omega\}$.

Definition 3.4 ([3] Quantitative Reactive Synthesis problem)**.** *Let* ϕ *be a temporal formula over* Σ*. The realizability problem for* ϕ *is the problem to find a strategy* g *such that* $[\![g, \phi]\!]$ *is maximal among all the strategies. We say that* ϕ *is realizable with value* T *if and only if there exists a strategy* g *such that* $[\![g, \phi]\!] \geq T$*.*

In order to distinguish the quantitative realizability from the standard one, we call the latter *qualitative realizability*. Unsurprisingly, the decidability and complexity of the quantitative realizability problem depends on the actual quantitative semantics we consider. As shown in [4], the quantitative realizability problem for $\text{LTL}[\mathscr{F}]$ is 2EXPTIME-complete. The same paper gives only an approximate solution for the logic $\text{LTL}^{\text{disc}}[\mathscr{D}]$, that allows to obtain a strategy that is optimal up to any desired accuracy $\epsilon > 0$. Later on in this paper, we propose a quantitative semantics to formalize ASAP properties, and we show that checking if a given strategy is optimal is 2EXPTIME for generic co-safety properties.

4. Optimal Model Checking

In this section, we formalize the problem of checking whether a controller satisfies optimally a temporal specification with respect to a given plant. To this purpose, we suppose to be given a quantitative semantics that assigns a real value $[\![S, \phi]\!]$ to a formula ϕ and a transition system S. Moreover, we define formally some notations and assume that the system $S = D \times P$ is the closed loop composition of a controller D and a plant P. Given a subset I of propositions in Σ, we say that an STS $S = \langle V, \Sigma, I, T \rangle$ is *input-enabled* with respect to I iff for all $s \in 2^V$ and $i \in 2^I$, there exist $o \in 2^{\Sigma \setminus I}$ and $s' \in 2^V$ such that $s, i \cdot o, s' \models T$. Given a subset O of propositions in Σ, we say S has a *functional dependency on the states* for O iff for every $s, s_1, s_2 \in 2^V, a_1, a_2 \in 2^\Sigma$, if $s, a_1, s_1 \models T$ and $s, a_2, s_2 \models T$, then $a_1(O) = a_2(O)$. The plant model is an STS P over Σ that is input-enabled with respect to C and has a functional dependency on the states for U, i.e. it is a symbolic representation of a set of strategies of the form $e : (2^C)^* \to 2^U$. The controller model is an STS D over Σ that is input-enabled with respect to U, i.e. it is a symbolic representation of a set of strategies of the form $g : (2^U)^+ \to 2^C$.

We are interested in proving that the closed loop $\mathscr{L}(D \times P)$ satisfies an LTL formula ϕ over Σ. This optimal model checking problem is formalized and adapted to our case as follows:

Definition 4.1 (Optimal model checking). *We say that D satisfies ϕ in the context P optimally, denoted by $D \models_*^P \phi$, iff there does not exist D' such that $[\![D' \times P, \phi]\!] > [\![D \times P, \phi]\!]$.*

Note that, in general, given two controllers D and D', their composition with the plant P may lead to completely different runs. Thus, it is not possible to compare the semantics of $D \times P$ and $D' \times P$ on single runs. This is why in Def. 4.1, we use $[\![D \times P, \phi]\!]$, which takes the minimum value of over all traces.

Definition 4.2 (Quantitative Realiability Under Assumptions). *Let ϕ be a temporal formula over the alphabet $\Sigma = C \cup U$ and P is a plant model. We say that ϕ is realizable under assumption P with value T if and only if there exists a controller D such that $[\![D \times P, \phi]\!] \geq T$.*

Suppose we can compute the value $T = [\![D \times P, \phi]\!]$, then we can reduce the optimal model checking problem to quantitative realizability of ϕ under the assumption P with a value $T' > T$. In the next sections, we will show how this reduction can rely on *qualitative* realizability in the particular case of the ASAP quantitative semantics.

5. ASAP LTL

In this section, we propose a new quantitative semantics that is suitable to formalize ASAP properties. Our definition of ASAP semantics gives a value in $\mathbb{N} \cup \{\infty\}$. We then cast the semantics to $[0, 1] \subset \mathbb{R}$ to be aligned with the standard definitions of quantitative semantics. Moreover, we introduce an alternative recursive definition of the ASAP semantics and show that the two definitions are equivalent.

The ASAP semantics is related to the notion of *informative prefix* introduced in [5] as the syntactical counterpart of the notion of "good prefix" for a co-safety formula: a finite computation that witnesses the satisfaction of the formula. While liveness, persistence, and reactivity LTL formulas do not necessarily have an informative prefix, if ϕ is a safety LTL formula then it is possible to build an automaton $A_{\neg\phi}$ that accepts exactly all informative prefixes of $\neg\phi$ [5]. This "tightness" results on the monitor for a formula can be extended to LTL_{EBR} by redefining the definition of informative prefix to include past operators as follows.

We say that a finite computation $\pi = \sigma_0\sigma_1 \ldots \sigma_{n-1}$ is an *informative prefix* for an LTL formula ϕ if and only if there exists a mapping $L : \{0, \ldots, n\} \mapsto 2^{cl(\phi)}$ that respects the following conditions:

1. $\phi \in L(0)$
2. $L(n)$ is empty.
3. For all $0 \leq i \leq n-1$ and $\psi \in L(i)$:
   - if ψ is $\top, p$, or $\neg p$, then it is satisfied by σ_i;
   - if $\psi = \psi_1 \wedge \psi_2$, then $\psi_1 \in L(i)$ and $\psi_2 \in L(i)$;
   - if $\psi = \psi_1 \vee \psi_2$, then $\psi_1 \in L(i)$ or $\psi_2 \in L(i)$;
   - if $\psi = \mathbf{X}\psi_1$, then $\psi_1 \in L(i+1)$;
   - if $\psi = \mathbf{Y}\psi_1$, then $i > 0$ and $\psi_1 \in L(i-1)$;
   - if $\psi = \mathbf{Z}\psi_1$, then $i = 0$ or $\psi_1 \in L(i-1)$;
   - if $\psi = \psi_1 \,\mathbf{U}\, \psi_2$, then $\psi_2 \in L(i)$ or [$\psi_1 \in L(i)$ and $\psi_1 \,\mathbf{U}\, \psi_2 \in L(i+1)$];
   - if $\psi = \psi_1 \mathbf{R} \psi_2$, then $\psi_2 \in L(i)$ and [$\psi_1 \in L(i)$ or $\psi_1 \mathbf{R} \psi_2 \in L(i+1)$];
   - if $\psi = \psi_1 \mathbf{S} \psi_2$, then $\psi_2 \in L(i)$ or [$\psi_1 \in L(i)$ and $\psi_1 \mathbf{S} \psi_2 \in L(i-1)$];
   - if $\psi = \psi_1 \mathbf{T} \psi_2$, then $\psi_2 \in L(i)$ and [$\psi_1 \in L(i)$ or $\psi_1 \mathbf{T} \psi_2 \in L(i-1)$].

The mapping L is called the *witness* for ϕ in π. Given an informative prefix π_n of a formula ϕ on an infinite trace σ, we say that π_n is minimal iff there does not exist any other informative prefix of σ shorter in length than π_n.

Definition 5.1 (ASAP semantics). *Let ϕ be an LTL formula and σ an infinite trace. The ASAP semantics of ϕ evaluated on σ, denoted by $[\![\sigma, \phi]\!]_{ASAP}$, is defined as follows: $[\![\sigma, \phi]\!]_{ASAP} = size(\pi_\sigma)$, where π_σ is the shortest informative prefix of σ.*

The ASAP semantics can also be defined by recursion on the structure of the formula.

Definition 5.2 (ASAP recursive semantics). *Let ϕ be an LTL formula and σ be a trace. The ASAP semantics of ϕ evaluated on σ at position i is a natural number n, denoted by $[\![\sigma, i, \phi]\!]^R_{ASAP}$ and defined recursively as follows:*

$$
\begin{aligned}
[\![\sigma, i, \top]\!]^R_{ASAP} &= 1 \\
[\![\sigma, i, \bot]\!]^R_{ASAP} &= +\infty \\
[\![\sigma, i, p]\!]^R_{ASAP} &= \begin{cases} 1 & \text{if } p \in \sigma_i \\ +\infty & \text{otherwise} \end{cases} \\
[\![\sigma, i, \neg p]\!]^R_{ASAP} &= \begin{cases} 1 & \text{if } p \notin \sigma_i \\ +\infty & \text{otherwise} \end{cases} \\
[\![\sigma, i, \phi_1 \wedge \phi_2]\!]^R_{ASAP} &= \max\left\{[\![\sigma, i, \phi_1]\!]^R_{ASAP}, [\![\sigma, i, \phi_2]\!]^R_{ASAP}\right\} \\
[\![\sigma, i, \phi_1 \vee \phi_2]\!]^R_{ASAP} &= \min\left\{[\![\sigma, i, \phi_1]\!]^R_{ASAP}, [\![\sigma, i, \phi_2]\!]^R_{ASAP}\right\} \\
[\![\sigma, i, \mathbf{X}\phi]\!]^R_{ASAP} &= [\![\sigma, i+1, \phi]\!]^R_{ASAP} + 1 \\
[\![\sigma, i, \mathbf{Y}\phi]\!]^R_{ASAP} &= \begin{cases} [\![\sigma, i-1, \phi]\!]^R_{ASAP} - 1 & \text{if } i > 0 \\ +\infty & \text{otherwise} \end{cases} \\
[\![\sigma, i, \mathbf{Z}\phi]\!]^R_{ASAP} &= \begin{cases} [\![\sigma, i-1, \phi]\!]^R_{ASAP} - 1 & \text{if } i > 0 \\ 1 & \text{otherwise} \end{cases} \\
[\![\sigma, i, \phi_1 \,\mathbf{U}\, \phi_2]\!]^R_{ASAP} &= \min_{k \geq 0}\left\{\max\left\{[\![\sigma, i+k, \phi_2]\!]^R_{ASAP} + k, \max_{0 \leq j < k}\left\{[\![\sigma, i+j, \phi_1]\!]^R_{ASAP} + j\right\}\right\}\right\} \\
[\![\sigma, i, \phi_1 \mathbf{R} \phi_2]\!]^R_{ASAP} &= \max_{k \geq 0}\left\{\min\left\{[\![\sigma, i+k, \phi_2]\!]^R_{ASAP} + k, \min_{0 \leq j < k}\left\{[\![\sigma, i+j, \phi_1]\!]^R_{ASAP} + j\right\}\right\}\right\} \\
[\![\sigma, i, \phi_1 \mathbf{S} \phi_2]\!]^R_{ASAP} &= \min_{0 \leq k \leq i}\left\{\max\left\{[\![\sigma, i-k, \phi_2]\!]^R_{ASAP} - k, \max_{0 < j \leq k}\left\{[\![\sigma, i-j, \phi_1]\!]^R_{ASAP} - j\right\}\right\}\right\} \\
[\![\sigma, i, \phi_1 \mathbf{T} \phi_2]\!]^R_{ASAP} &= \max_{0 \leq k \leq i}\left\{\min\left\{[\![\sigma, i-k, \phi_2]\!]^R_{ASAP} - k, \min_{0 \leq j < k}\left\{[\![\sigma, i-j, \phi_1]\!]^R_{ASAP} - j\right\}\right\}\right\}
\end{aligned}
$$

where $max, min, +$, and $-$ are defined on the extended natural numbers $\mathbb{N} \cup \{\infty\}$. We recall that $\infty - k = \infty$, if $k \in \mathbb{N}$, $0 - k = 0$, if $k \in \mathbb{N} \cup \{\infty\}$, and $k - \infty = 0$, if $k \in \mathbb{N}$. We say that $[\![\sigma, \phi]\!]^R_{ASAP} = n$ if and only if $[\![\sigma, 0, \phi]\!]^R_{ASAP} = n$.

We now prove that the two quantitative semantics are equivalent.

Theorem 5.1. *Let ϕ be an LTL formula, $\sigma = \sigma_0\sigma_1 \ldots$ be an infinite trace, and $\pi_\sigma = \sigma_0\sigma_1 \ldots \sigma_{n-1}$ be the prefix of σ of length n. If π_σ is a minimal informative prefix for ϕ, then $[\![\sigma, \phi]\!]^R_{ASAP} = [\![\sigma, \phi]\!]_{ASAP}$.*

Finally, we define the normalized version of the ASAP semantics as follows:

Definition 5.3 (Normalized ASAP Semantics). *For any formula ϕ and trace σ,*

$$
[\![\sigma, \phi]\!]^N_{ASAP} = \begin{cases} 0 & \text{if } [\![\sigma, \phi]\!]_{ASAP} = +\infty \\ 1/([\![\sigma, \phi]\!]_{ASAP}) & \text{otherwise} \end{cases}
$$

Thus, $[\![\sigma, \phi]\!]^N_{ASAP}$ tends to 0 while $[\![\sigma, \phi]\!]_{ASAP}$ tends to $+\infty$.

6. Reduction of ASAP LTL Model Checking to Reactive Synthesis

We now show that, in the case of co-safety formulas, the optimal model checking problem with ASAP LTL semantics can be reduced to a reachability game. The key step is to build an arena A_ϕ^P with a parametrized winning condition $\gamma(k)$ so that there exists a winning strategy iff there exists a controller D such that $[\![D \times P, \phi]\!]_{\text{ASAP}} < k$. With such a construction, we can reduce the optimal model checking problem to a qualitative synthesis problem. This is achieved by 1) computing $k = [\![D \times P, \phi]\!]_{\text{ASAP}}$ and 2) checking if there exists a winning strategy for the game A_ϕ^P with winning condition $\gamma(k)$.

We first define an STS counting the transitions from the initial state. It has a new state variable $steps$ so that initially $steps$ has value 0 and at each step it is increased by 1. Formally, we define the STS A_{steps} as follows: $A_{steps} := \langle \{steps\}, \emptyset, steps = 0, steps' \leftrightarrow ite(steps < max, steps + 1, steps) \rangle$, where *ite* is the Boolean encoding of an if-then-else term and *max* is a sufficiently large constant. Let A_ϕ be a deterministic STS with a boolean condition f_ϕ on the STS variables that is reached by any informative prefix of ϕ and built starting from an LTL_{EBR} formula, as given by [1]. Recall that a boolean condition is reached by a STS S iff there exists a finite path $s_0 \dots s_k$ in S such that $s_k \models f_\phi$. Step 1, i.e. computing k, can be solved by looking for a minimal m such that $D \times P \times A_\phi \times A_{steps} \models \mathbf{F}(f_\phi \wedge steps \leq m)$. This can be achieved with an incremental number of model checking problems or with parametric model checking.

Let A_P be the deterministic STS accepting the same language of P with error state e_P that is reached wherever the trace is not accepted by P. Then we define the arena A_ϕ^P as the product $A_P \times A_\phi \times A_s^c$ with the winning condition $\gamma(k)$ defined as $e_P \vee (f_\phi \wedge step < k)$.

A winning strategy for the reachability game with winning condition $\gamma(k)$ and arena A_ϕ^P guarantees that in every play of the game the controller can reach a state where either the plant monitor is in e_P (the play is not a valid trace of the plant), or the monitor for ϕ is in the accepting state and the value of $step$ is less than the bound k (formula ϕ is true in less than k steps). We can prove that existence of a winning strategy corresponds to existence of a controller that can enforce ϕ with cost less than k.

Theorem 6.1. *There exists a winning strategy for the reachability game with winning condition $\gamma(k)$ and arena A_ϕ^P iff there exists a controller D such that $[\![D \times P, \phi]\!]_{\text{ASAP}} < k$.*

The next theorem states the complexity of the procedure.

Theorem 6.2. *The ASAP LTL model checking problem for co-LTL_{EBR} is 2EXPTIME-complete.*

Remark. *It is not possible to synthesize a controller winning the game enforcing the monitor plant into error state, since the plant is assumed to be input-enabled and so the built monitor accepts any possible combination of controllable variables. Therefore, it is up to the environment to choose whether it goes into error state or not.*

7. Related work

The problem of optimizing winning strategies has been studied since the early 2000 [6]. Earlier works addressed optimization of the memory size of the controller (i.e., number of states of the automaton) [7] and minimization of waiting times for request-response conditions [8]. In this paper, we relate the problem of model checking open systems, also studied in [9], to the problem of optimal synthesis, using a quantitative semantics for LTL to define the problem of optimal model checking in terms of synthesis of a better strategy.

Several formalisms for quantitative semantics for temporal logic exist in the literature [10, 11, 12, 13, 14, 15, 16]. While these formalisms can be used to define optimality criteria, most of the current works concentrate on the model checking problem and do not consider the synthesis problem. Notable exceptions are the logics $\text{LTL}[\mathscr{F}]$ and $\text{LTL}^{\text{disc}}[\mathscr{D}]$, introduced in [4], which extend LTL, the first, with propositional quality operators to prioritize and weight different satisfaction possibilities and, the second, with discounting operators, to take into account the delay incurred in the satisfaction of

eventualities. The logic LTL$[\mathscr{F}]$ has been subsequently used to develop algorithms for quantitative assume-guarantee synthesis of GR(1) properties [3], for compositional assume-guarantee synthesis [17], and for the synthesis of finite-memory controllers of discrete event systems [18]. In particular, the ASAP quantitative semantics is very related to the logic LTL$^{\text{disc}}[\mathscr{D}]$, which extends LTL with a "discounted until" $\mathbf{U}_\eta$ operator that takes into consideration the length of the prefix needed to satisfy the eventualities. It is parametrized by a function η over the natural numbers. Indeed, with discounting function $\eta_U(i) = \frac{1}{i+1}$, if p and q are propositional, the semantics of $p\,\mathbf{U}_\eta q$ is the same as the ASAP semantics of $p\,\mathbf{U}\,q$. In [4], it is also remarked that a discounted next can be expressed using the normal $\mathbf{X}$ and the discounted until: one has to define $\eta_X(i)$ such that $\eta_X(i) = \eta_U(i+1)$ and then define the discounted next $\mathbf{X}_\eta(\psi)$ as $\mathbf{X}(\bot\,\mathbf{U}_{\eta_X}\psi)$. However, in this case, the semantics of the discounted next and the discounted until use different discount functions. Moreover, the recursive definition of the temporal operators are not valid for LTL$^{\text{disc}}[\mathscr{D}]$. Thus, although the motivations are the same, the quantitative semantics that we propose differ from LTL$^{\text{disc}}[\mathscr{D}]$ and is more natural to express ASAP requirements in the sense that it is directly connected to the length of informative prefixes that satisfy the formula.

A line of research related to this paper is the study of "good enough/best effort synthesis", a variant of synthesis in which the system is required to satisfy the specification only as long and as much as allowed by the environment, and it is allowed to fail when a satisfying computation does not exist [19, 20, 21, 22, 23]. While the formalisms and the algorithms are related to our work, the setting is quite different, as they aim to synthesize a controller that is "good enough" to at least partially respect the desired properties when they cannot be enforced, and not to synthesize the "best controller" under some optimality criteria.

8. Proof of Concept Evaluation

We implemented the optimal model checking procedure with a toolchain called `optimal-strategy` that uses the nuXmv model checker [2] to build the monitor for the formula, solve the parametric model checking, and build the arena for the reachability game. The game is solved by the symbolic safety game solver `simple-synth` we developed from scratch. The game arena is produced in AIGER format [24], to allow using any tool for reactive synthesis for solving the game. All code, examples and benchmarks to replicate the experiments are available in the artifact [1]. The toolchain can either solve the parameterized model checking defined in Section 5 to compute the upper-bound k for the reachability game, or start with a user-defined upper bound.

As a first benchmark, we tested the fox, dog and chicken example introduced in Section 2 and we have proved that the given controller is not optimal, since it enforces the specification in 15 steps, while the toolchain synthesized a controller which satisfies the specification in 7 steps.

To test the tool, we considered scalable formulas of increasing size. Since there are no other tools solving the optimal model checking problems for ASAP LTL, there are no comparisons with other software. The first formula considers a scenario where a car moves on a $n \times n$ square grid. The initial controller can only move the car up, down, left and right. The optimal controller synthesized by the tool is allowed also to move diagonally. We require the controller to reach n points along the diagonal of the grid, avoiding a set bad of bad states, as formalized by:

$$\neg bad\,\mathbf{U}\,(s_{n-1} \wedge \mathbf{O}(s_{n-2} \wedge \mathbf{O}(s_{n-3} \wedge \mathbf{O}(\ldots \wedge \mathbf{O}s_0)))), \tag{1}$$

where bad is the set of bad states defined as follows: $bad = \bigvee_{i=0}^{\lfloor n-1 \rfloor}(x = i \wedge y = i+1)$.

The other 6 categories consist of synthetic formulas which are conjunctions and disjunctions of next, bounded finally, bounded globally, and unbounded finally and until operators. The plant consists of n controllable variables c_i, n uncontrollable variables u_i and is defined to flip the values of uncontrollable variables at each step, forcing mutual exclusion between odd and even uncontrollable variable. The formulas

[1] https://drive.google.com/file/d/1qbs6FvfcFeUQitwFWGImfuQcjgOkzQ7I/view?usp=sharing

are aimed to test the toolchain by increasing the number of variables and the nesting of next operators:

$$\bigwedge_{i=0}^{n-1} \mathbf{X}^i(\mathbf{F}^{[0,6]}(c_i = u_i)) \quad (2) \qquad \bigwedge_{i=0}^{n-1} \mathbf{X}^i\mathbf{F}(c_i = u_i) \quad (5)$$

$$\bigwedge_{i=0}^{n-1} (\mathbf{X}^i\mathbf{G}^{[0,3]}\mathbf{F}^{[0,3]}(c = u_i)) \quad (3) \qquad \bigwedge_{i=0}^{n-1} \mathbf{X}^i((\neg c)\,\mathbf{U}\,(c = u_i)) \quad (6)$$

$$\bigwedge_{i=0}^{n-1} \mathbf{X}^{2i}(\mathbf{F}^{[0,6]}(c = u_0) \wedge \mathbf{X}\mathbf{F}^{[0,6]}(c \neq u_1)) \quad (4) \qquad \bigwedge_{i=0}^{n-1} \mathbf{X}^{2i}(\mathbf{F}(c = u_0) \wedge \mathbf{X}\mathbf{F}(c \neq u_1)) \quad (7)$$

We complemented the scalable formulas with 38 different co-safety formulas of size ranging from 2 to 179, paired with different plants. The minimum, maximum and average running time were respectively 88 ms, 2332345 ms and 65291.2 ms.

Overall, the toolchain is efficient for small or easy to solve formulas, and it scales with the complexity of the reactive synthesis phase.

9. Conclusions and Future Work

In this paper, we formalized the requirements of fulfilling some temporal reachability properties ASAP for a controller assuming some constraints on the environment. In order to do that, we formalized the optimal model checking problem w.r.t. LTL with a quantitative semantics able to capture the time needed to fulfill a property and so that its optimal model checking problem corresponds to check the ASAP requirement. Finally, we provided a reduction to standard co-safety synthesis. A proof-of-concept evaluation shows that the approach is indeed feasible. The paper opens various problems left for future work, e.g., how to solve the optimal model checking with other semantics, how to generalize the reduction to synthesis to support larger fragments of ASAP LTL, and how to scale up the verification for example exploiting incremental approaches.

Acknowledgments: The authors want to thank all the anonymous reviewers of OVERLAY 2024 for the insightful comments on a preliminary version of this paper.

References

[1] Alessandro Cimatti, Luca Geatti, Nicola Gigante, Angelo Montanari, and Stefano Tonetta. Extended bounded response LTL: a new safety fragment for efficient reactive synthesis. *Form Methods Syst Des*, 2021.

[2] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. The nuXmv Symbolic Model Checker. In *CAV*, volume 8559 of *LNCS*, pages 334–342. Springer, 2014.

[3] Shaull Almagor, Orna Kupferman, Jan Oliver Ringert, and Yaron Velner. Quantitative Assume Guarantee Synthesis. In *CAV*, volume 10427, pages 353–374, 2017.

[4] Shaull Almagor, Udi Boker, and Orna Kupferman. Formally reasoning about quality. *J. ACM*, 63(3), 06 2016.

[5] Orna Kupferman and Moshe Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19, 1999.

[6] Wolfgang Thomas. Optimizing winning strategies in regular infinite games. In *SOFSEM*, volume 4910 of *LNCS*, pages 118–123. Springer, 2008.

[7] Michael Holtmann and Christof Löding. Memory Reduction for Strategies in Infinite Games. In *CIAA*, volume 4783 of *LNCS*, pages 253–264. Springer, 2007.

[8] Florian Horn, Wolfgang Thomas, Nico Wallmeier, and Martin Zimmermann. Optimal strategy synthesis for request-response games. *RAIRO Theor. Informatics Appl.*, 49(3):179–203, 2015.

[9] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. Module Checking. *Inf. Comput.*, 164(2):322–344, 2001.

[10] Tzanis Anevlavis, Matthew Philippe, Daniel Neider, and Paulo Tabuada. Being correct is not enough: Efficient verification using robust linear temporal logic. *ACM Trans. Comput. Logic*, 23(2), 2022.

[11] Patricia Bouyer, Nicolas Markey, and Raj Mohan Matteplackel. Averaging in LTL. In *CONCUR*, volume 8704 of *LNCS*, pages 266–280. Springer, 2014.

[12] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4):23:1–23:38, 2010.

[13] Manfred Droste, Gustav Grabolle, and George Rahonis. Weighted Linear Dynamic Logic. *Int. J. Found. Comput. Sci.*, 35(1&2):145–177, 2024.

[14] Marco Faella, Axel Legay, and Mariëlle Stoelinga. Model Checking Quantitative Linear Time Logic. In *QAPL*, number 3 in Electronic Notes in Theoretical Computer Science, pages 61–77. Elsevier, 2008.

[15] Paul Gastin and Benjamin Monmege. A unifying survey on weighted logics and weighted automata. *Soft Comput.*, 22(4):1047–1065, 2018.

[16] Yongming Li, Manfred Droste, and Lihui Lei. Model checking of linear-time properties in multi-valued systems. *Inf. Sci.*, 377:51–74, 2017.

[17] Rafael Dewes and Rayna Dimitrova. Compositional high-quality synthesis. In *Automated Technology for Verification and Analysis*, pages 334–354. Springer Nature Switzerland, 2023.

[18] Ami Sakakibara, Natsuki Urabe, and Toshimitsu Ushio. Finite-Memory Supervisory Control of Discrete Event Systems for LTL[$\mathscr{F}$] Specifications. *IEEE Transactions on Automatic Control*, 67(12):6896–6903, 2022.

[19] Shaull Almagor and Orna Kupferman. Good-enough synthesis. In *CAV*, pages 541–563, 2020.

[20] Benjamin Aminof, Giuseppe De Giacomo, and Sasha Rubin. Best-Effort Synthesis: Doing Your Best Is Not Harder Than Giving Up. In *IJCAI*, pages 1766–1772, 2021.

[21] Benjamin Aminof, Giuseppe De Giacomo, and Sasha Rubin. Reactive Synthesis of Dominant Strategies. In *AAAI*, pages 6228–6235, 2023.

[22] Benjamin Aminof, Giuseppe De Giacomo, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. Synthesizing Best-effort Strategies under Multiple Environment Specifications. In *KR*, pages 42–51, 2021.

[23] Morteza Lahijanian, Shaull Almagor, Dror Fried, Lydia E. Kavraki, and Moshe Y. Vardi. This Time the Robot Settles for a Cost: A Quantitative Approach to Temporal Logic Planning with Partial Satisfaction. In *AAAI*, pages 3664–3671, 2015.

[24] Armin Biere, Keijo Heljanko, and Siert Wieringa. AIGER 1.9 and beyond. Technical Report 11/2, Institute for Formal Models and Verification, Johannes Kepler University, 2011.