Auton Agent Multi-Agent Syst (2025) 39:31
https://doi.org/10.1007/s10458-025-09712-w

Behavioral QLTL

Giuseppe De Giacomo[1,2] · **Giuseppe Perelli**[3]

Accepted: 6 June 2025

Abstract
This paper introduces Behavioral QLTL, a "behavioral" variant of Linear Temporal Logic (LTL) with second-order quantifiers. Behavioral QLTL is characterized by the fact that the functions that assign the truth value of the quantified propositions along the trace can only depend on the past. In other words, such functions must be "processes" (Abadi et al., Realizable and Unrealizable Specifications of Reactive Systems, 1989) . This gives the logic a strategic flavor that we usually associate with planning. Indeed we show that temporally extended planning in nondeterministic domains and ltl synthesis are expressed in Behavioral QLTL through formulas with a simple quantification alternation. As such alternation increases, we get to forms of planning/synthesis in which contingent and conformant planning aspects get mixed. We study this logic from the computational point of view and compare it to the original QLTL (with non-behavioral semantics) and simpler forms of behavioral semantics.

Keywords Strategic reasoning · Behavioral semantics · Foundations of planning · Reasoning about actions

1 Introduction

Since the very early time of AI, researchers have tried to reduce planning to logical reasoning, i.e., satisfiability, validity, logical implication [2]. However as we consider more and more sophisticated forms of planning this becomes more and more challenging, because the logical reasoning required quickly becomes second-order. One prominent case is if we want to express the model of the world (aka the environment) and the goal of the agent directly in Lin-

✉ Giuseppe Perelli
perelli@di.uniroma1.it

[1] Department of Computer Science, University of Oxford, Oxford, UK

[2] Department of Computer, Automated, and Business Engineering, Sapienza University of Rome, Rome, Italy

[3] Department of Computer Science, Sapienza University of Rome, Rome, Italy

Published online: 30 June 2025

ear Temporal Logic (LTL). LTL has been often adopted also in Artificial Intelligence. Examples are the pioneering work on using temporal logic as a sort of programming language through the MetateM framework [3], the work on temporally extended goals and declarative control constraints [4, 5], the work on planning via model-checking [6–9], the work on adopting LTL logical reasoning (plus some meta-theoretic manipulation) for certain forms of planning [10, 11]. More recently the connection between planning in nondeterministic domains and (reactive) synthesis [12] has been investigated, and in fact it has been shown that planning in nondeterministic domains can be seen in general terms as a form of synthesis in presence of a model of the environment [13, 14], also related to synthesis under assumptions [15, 16].

However the connection between planning and synthesis also clarifies formally that we cannot use directly the standard forms of reasoning in LTL, such as satisfiability, validity, or logical implication, to do planning. Indeed the logical reasoning task we have to adopt is a nonstandard one, called "*realizability*" [12, 17], which is inherently a second-order form of reasoning on LTL specifications.

So one question comes natural: can we use the second-order version of LTL, called QLTL (or QPTL) [18] and then use its classic reasoning tasks, such as satisfiability, validity and logical implication, to capture planning and synthesis?

In [11] a positive answer was given limited to conformant planning [19], in which we have partial observability on the environment and, in particular, we cannot fully observe the initial state and the environment response to agent actions, which however are deterministic. Hence, in conformant planning we need to synthesize plans/strategies that work (in the deterministic domain) in spite of the lack of knowledge. Calvanese et al. [11] shows that exploiting existential and universal quantifications, to account for the lack of knowledge, QLTL could actually capture conformant planning through standard satisfiability.

However, the results there do not apply when the environment is nondeterministic, as in contingent planning (with or without full observability) [19].

The reason for this is very profound. Any plan/strategy must be a "*process*", i.e., a function that observes what has happened so far (the history), observes the current state, and takes a decision (conditional on what observed) on the next action to do [1]. QLTL instead interprets quantified propositions (i.e., in the case of planning, the actions to be chosen) through functions that have access to the whole traces, i.e., also the future instants, hence they cannot be considered processes. This is a clear mismatch that makes standard QLTL unsuitable to capture planning through standard reasoning tasks.

This mismatch is not only a characteristic of QLTL, but, interestingly, even of logics that have been introduced specifically for strategic reasoning such as Strategy Logic (SL) [20, 21]. This has led to investigating the "*behavioral*" semantics in these logics, i.e., a semantics based on processes. In their seminal work [21], Mogavero et al. introduce and analyze the behavioral aspects of quantification in SL: a logic for reasoning about the strategic behavior of agents in a context where the properties of executions are expressed in LTL. They show that restricting to behavioral quantification of strategies is a way of both making the semantics more realistic and computationally easier. In addition, they proved that behavioral and non-behavioral semantics coincide for certain fragments, including the one corresponding to the well known ATLS* [22], but diverge for more interesting classes of formulas, e.g., the ones that can express game-theoretic properties such as Nash Equilibria and the like. This has started a new line of research that aims at identifying new notions of behavioral and non-behavioral quantification, as well as characterize the syntactic fragments that are invariant to these semantic variations [23–25].

In this paper, inspired by the study of behavioral semantics in Strategy Logic, we introduce a simple and elegant variant of QLTL with a behavioral semantics. The resulting logic, called *Behavioral*-QLTL (QLTL_B), maintains the same syntax of QLTL, but is characterized by the fact that the functions that assign the truth value of the quantified propositions along the trace can only depend on the past. In other words such functions must indeed be "*processes*". This makes QLTL_B perfectly suitable to capture extended forms of planning and synthesis through standard reasoning tasks (satisfiability in particular).

In QLTL_B, planning for temporally extended goals in nondeterministic domains, as well as LTL synthesis, are expressed through formulas with a simple quantification alternation.

While, as this alternation increases, we get to forms of planning/synthesis in which contingent and conformant planning aspects get mixed by controlling via quantification what is visible of the current history to take a decision on.

For example, the QLTL_B formula of the form $\exists Y \forall X \psi$ represents the conformant planning over the LTL specification (of both environment model and goal) ψ, as it is intended in [19]. Here we use $\forall X$ to hide in the history the propositions (a.k.a. *fluents*) that are not visible to the agent. Note that this could be done also with standard QLTL, since $\exists Y$ is put upfront as it cannot depend on the nondeterministic evolution of X. The QLTL_B formula $\forall X \exists Y \psi$ represents contingent planning in fully observable domains [19], also known as *Strong Planning in Fully Observable Nondeterministic Domains* (FOND) [26, 27], as well as LTL synthesis [12]. The QLTL_B formula $\forall X_1 \exists Y \forall X_2 \varphi$ represents the problem of contingent planning under partial observability [19], also known as *Strong Planning in Partially Observable Nondeterministic Domains* (POND) [27]. Here, X_1 and X_2 are, respectively, the visible and hidden propositions controlled by the environment and the strategy corresponding to the Skolem function assigning the values to Y depends on the values of X_1 in the history so far but not on the values of X_2, which indeed remain non-observable to the agent. By going even further in alternation, we get a generalization of POND where a number the controllable variables of the agent depend individually on more and more environment variables. In other words, we have a hierarchy of partial observability over the whole history on which the various variable under the control of the agent can depend upon. Interestingly, if we consider the agent controlled variables as independent actuators, then this instantiates the problem of distributed synthesis with strictly decreasing levels of information studied in formal methods [28–30].

We study QLTL_B by introducing a formal semantics that is *Skolem-based*, meaning that we assign existential values through Skolem-like functions that depend on the universal (adversarial) choice of the variables of interest. Specifically we restrict such Skolem function to depend only on the past and hence behave as processes/strategies/plans. As a matter of fact, such Skolem functions can be represented as suitable labeled trees, describing all the possible executions of a given process that receive inputs from the environment.

We then study satisfiability in QLTL_B and characterize its complexity as $(n+1)$-EXPTIME-complete, with n being the number of quantification blocks of the form $\forall X_i \exists Y_i$ in the formula.

Note that this is substantially lower than the complexity of satisfiability for classic QLTL, which depends on the overall quantifier alternation in the formula, and in particular is $2(n-1)$-EXSPACE-complete.

Interestingly, instantiating our satisfiability procedure we get an optimal technique for solving synthesis, and planning in nondeterministic domains, for LTL goals in the case of full

observability and partial observability. Indeed, both the formula $\forall X \exists Y \psi$ for the case of full observability and the formula $\forall X_1 \exists Y \forall X_2 \varphi$ for the case of partial observability, include a single block of the form $\forall X_i \exists Y_i$, and hence satisfiability can be checked in 2-EXPTIME, thus matching the 2-EXPTIME-completeness of the two problems [12, 31].

In addition to QLTL$_{\mathrm{B}}$, we consider a weak variant, called *Weak Behavioral*-QLTL (QLTL$_{\mathrm{WB}}$), where the history is always visible but we have restriction visibility on the current instant of the computation. We show that the complexity of satisfiability in QLTL$_{\mathrm{WB}}$ is 2-EXPTIME-complete, regardless of the number and alternation of quantifiers, as it is the case for SL [21]. The reason for this is that processes are modeled in a way that they have full visibility on the past computation. This allows them to find plans/strategies by means of a local reasoning, and so without employing computationally expensive automata projections. As for the case of QLTL$_{\mathrm{B}}$, such procedure is optimal to solve the corresponding synthesis problems, as the matching lower-bound is again provided by a reduction of them. Interestingly the QLTL$_{\mathrm{WB}}$ can be seen as a behavioral QLTL closer to the behavioral variants of *Strategy Logic* studied in [21, 32], which we discuss later in the paper.

In Table 1 we report the complexity results of the Quantified LTL, introduced in [33] and the two behavioral variants introduced here.

Outline The paper is structured as follows. In Section 2 we recall the main definitions and results about LTL and QLTL. In Section 3 we introduce a Skolem-based semantics for QLTL (still with the non-behavioral semantics), which we prove to be equivalent to the standard one. In Section 4 we analyze the behavioral variation of QLTL, namely, *Behavioral* QLTL (QLTL$_{\mathrm{B}}$), based on suitable restrictions of the Skolem functions in place. In Section 6 we study some technical properties of the logic. In Section 7 we study satifability, specifically we present an automata based technique for checking satisfiability, and give a computational complexity characterization of the problem. In Section 8, we turn to *Weak-behavioral* QLTL (QLTL$_{\mathrm{WB}}$) and study satisfiability in this logic. In Section 9 we discuss the relationship between behavioral variants of QLTL and the behavioral variants of Strategy Logic, in particular we show that the simpler QLTL$_{\mathrm{WB}}$(and not QLTL$_{\mathrm{B}}$) is the counterpart in QLTL of the behavioral semantics typically adopted in Strategy Logic. Finally, in Section 10 we give some concluding remarks.

This paper is a significant extension of [34]. Here, we provide full details of the proofs for QLTL$_{\mathrm{B}}$. In addition, we provide two more sections, namely Section 8 and Section 9.

2 Quantified linear temporal logic

We introduce *Quantified Linear Temporal Logic* as an extension of *Linear Temporal Logic*.

Table 1 Summary of main complexity results. Parameter n refers to the number of quantification blocks

Language	Complexity
Quantified LTL (QLTL) [33]	$2(n-1)$-EXPSPACE-complete [18, 33]
Behavioral QLTL (QLTL$_{\mathrm{B}}$)	$(n+1)$-EXPTIME-complete (Thm 7)
Weak-Behavioral QLTL (QLTL$_{\mathrm{WB}}$)	2-EXPTIME-complete (Thm 11)

Linear Temporal Logic Linear Temporal Logic (LTL) was originally proposed in Computer Science as a specification language for concurrent programs [35]. Formulas of LTL are built from a set Var of *propositional variables* (or simply variables), together with Boolean and temporal operators. Its syntax can be described as follows:

$$\psi ::= x \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \mathrm{X}\psi \mid \psi \mathrm{U} \psi$$

where $x \in \mathrm{Var}$ is a propositional variable.

Intuitively, the formula $\mathrm{X}\psi$ says that ψ holds at the *next* instant. Moreover, the formula $\psi_1 \mathrm{U} \psi_2$ says that at some future instant ψ_2 holds and *until* that point, ψ_1 holds.

We also use the standard Boolean abbreviations $\mathrm{true} := x \vee \neg x$(*true*), $\mathrm{false} := \neg\mathrm{true}$(*false*), and $\psi_1 \rightarrow \psi_2 := \neg\psi_1 \vee \psi_2$(*implication*). In addition, we also use the binary operator $\psi_1 \mathrm{R} \psi_2 \doteq \neg(\neg\psi_1 \mathrm{U} \neg\psi_2)$(*release*) and the unary operators $\mathrm{F}\psi := \mathrm{true}\mathrm{U}\psi$(*eventually*) and $\mathrm{G}\psi := \neg\mathrm{F}\neg\psi$(*globally*).

The classic semantics of LTL is given in terms of infinite traces, i.e., truth-value assignments over the natural numbers. More precisely, a *trace* $\pi \in (2^{\mathrm{Var}})^{\omega}$ is an infinite sequence of truth assignments over the set of variables Var, where $(\cdot)^{\omega}$ is the classic omega operator used to denote such infinite sequences. By $\pi(i) \in 2^{\mathrm{Var}}$, we denote the *i*-th truth assignment of the infinite sequence π. Along the paper, we might refer to finite *segments* of a computation π. More precisely, for two indexes $i, j \in \mathbb{N}$, by $\pi(i, j) \doteq \pi(i), \ldots, \pi(j) \in (2^{\mathrm{Var}})^{*}$ we denote the finite segment of π from it's *i*-th to its *j*-th position, where $(\cdot)^{*}$ is the classic Kleene's star used to denote finite sequences of any length. A segment $\pi(0, j)$ starting from 0 is also called a *prefix* and is sometimes denoted $\pi_{\leq j}$. Moreover, we sometimes use π_X to denote a trace over a subset $X \subseteq \mathrm{Var}$ of variables, that is, we make explicit the range of variables on which the trace is defined. Also, we may use the notation π_{-X} for a trace over the subset $\mathrm{Var} \setminus X$, whenever the set Var is clear from the context.

We say that an LTL formula ψ is true on an assignment π at instant *i*, written $\pi, i \models_{\mathrm{LTL}} \psi$, if:

- $\pi, i \models_{\mathrm{LTL}} x$, for $x \in \mathrm{Var}$ iff $x \in \pi(i)$;
- $\pi, i \models_{\mathrm{LTL}} \neg\psi$ iff $\pi, i \not\models_{\mathrm{LTL}} \psi$;
- $\pi, i \models_{\mathrm{LTL}} \psi_1 \vee \psi_2$ iff either $\pi, i \models_{\mathrm{LTL}} \psi_1$ or $\pi, i \models_{\mathrm{LTL}} \psi_2$;
- $\pi, i \models_{\mathrm{LTL}} \psi_1 \wedge \psi_2$ iff both $\pi, i \models_{\mathrm{LTL}} \psi_1$ and $\pi, i \models_{\mathrm{LTL}} \psi_2$;
- $\pi, i \models_{\mathrm{LTL}} \mathrm{X}\psi$ iff $\pi, i+1 \models_{\mathrm{LTL}} \psi$;
- $\pi, i \models_{\mathrm{LTL}} \psi_1 \mathrm{U} \psi_2$ iff for some $j \geq i$, we have that $\pi, j \models_{\mathrm{LTL}} \psi_2$ and for all $k \in \{i, \ldots j-1\}$, we have that $\pi, k \models_{\mathrm{LTL}} \psi_1$.

A formula ψ is *true* over π, written $\pi \models_{\mathrm{LTL}} \psi$, iff $\pi, 0 \models_{\mathrm{LTL}} \psi$. A formula ψ is *satisfiable* if it is true on some trace and *valid* if it is true in every trace.

Quantified linear-time temporal logic Quantified Linear-Temporal Logic (QLTL) is an extension of LTL with two *Second-order* quantifiers [18]. Its formulas are built using the classic LTL Boolean and temporal operators, on top of which existential and universal quantification over variables is applied. Formally, the syntax is given as follows:

$$\varphi ::= \exists x \varphi \mid \forall x \varphi \mid x \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi,$$

where $x \in \mathrm{Var}$ is a propositional variable.

Note that this is a proper extension of LTL, as QLTL has the same expressive power of MSO [18], whereas LTL is equivalent to FOL [36].

In order to define the semantics of QLTL, we introduce some notation.

For a trace π and a set of variables $X \subseteq \mathrm{Var}$, by $\mathrm{Prj}(\pi, X)$ we denote the *projection* trace over X defined as $\mathrm{Prj}(\pi, X)(i) \doteq \pi(i) \cap X$ at any time point $i \in \mathbb{N}$. Moreover, by $\mathrm{Prj}(\pi, -X) \doteq \mathrm{Prj}(\pi, \mathrm{Var} \setminus X)$ we denote the projection trace over the complement of X. For a single variable x, we simplify the notation as $\mathrm{Prj}(\pi, x) \doteq \mathrm{Prj}(\pi, \{x\})$ and $\mathrm{Prj}(\pi, -x) \doteq \mathrm{Prj}(\pi, \mathrm{Var} \setminus \{x\})$. Finally, we say that π and π' *agree* over X if $\mathrm{Prj}(\pi, X) = \mathrm{Prj}(\pi', X)$.

Observe that we can reverse the projection operation by combining traces over disjoint sets of variables. More formally, for two disjoint sets $X, X' \subseteq \mathrm{Var}$ and two traces π_X and $\pi_{X'}$ over X and X', respectively, we define the combined trace $\pi_X \Cup \pi_{X'}$ as the (unique) trace over $X \cup X'$ such that its projections on X and X' correspond to π_X and $\pi_{X'}$, respectively.

The *classic* semantics of the quantifiers in a QLTL formula φ over a trace π, at instant i, denoted $\pi, i \models_{\mathrm{C}} \varphi$, is defined as follows:

- $\pi, i \models_{\mathrm{C}} \psi$ iff $\pi, i \models_{\mathrm{LTL}} \psi$ for every quantifier-free (LTL) formula ψ;
- $\pi, i \models_{\mathrm{C}} \exists x \varphi$ iff there is a trace π' that agrees with π over $-x$ such that $\pi', i \models_{\mathrm{C}} \varphi$;
- $\pi, i \models_{\mathrm{C}} \forall x \varphi$ iff for each trace π' that agrees with π over $-x$, it holds that $\pi', i \models_{\mathrm{C}} \varphi$;

A variable x is *free* in φ if it occurs at least once out of the scope of either $\exists x$ or $\forall x$ in φ. By $\mathrm{free}(\varphi)$ we denote the set of free variables in φ.

As for LTL, we say that φ is true on π, and write $\pi \models_{\mathrm{C}} \varphi$ iff $\pi, 0 \models_{\mathrm{C}} \varphi$. Analogously, a formula φ is *satisfiable* if it is true on some trace π, whereas it is *valid* if it is true on every possible trace π. Note that, as quantifications in the formula replace the trace over the variables in their scope, we can assume that π are traces over the set $\mathrm{free}(\varphi)$ of free variables in φ.

For convenience, and without loss of generality, QLTL is typically used in *prenex normal form*, i.e., according to the following syntax:

$$\varphi ::= \exists x \varphi \mid \forall x \varphi \mid \psi$$

where ψ is an LTL formula over the the propositional variables Var.

Hence a QLTL formula in *prenex normal form* has the form $\wp\psi$, where $\wp = \mathrm{Qn}_1 x_1 \ldots \mathrm{Qn}_n x_n$ is a *prefix quantification* with $\mathrm{Qn}_i \in \{\exists, \forall\}$ and x_i being a variable occurring on a *quantifier-free* subformula ψ. Every QLTL formula can be rewritten in prenex normal form, meaning that such rewriting is true on the same set of traces. Consider for instance the formula $\mathrm{G}(\exists y(y \wedge \mathrm{X}\neg y))$. This is equivalent to $\forall x \exists y(\mathrm{singleton}(x) \rightarrow (\mathrm{G}(x \rightarrow (y \wedge \mathrm{X}\neg y))))$, with $\mathrm{singleton}(x) \doteq \mathrm{F}x \wedge \mathrm{G}(x \rightarrow \mathrm{XG}\neg x)$ expressing the fact that x is true exactly once on the trace [1]. A full proof of the reduction to prenex normal form can be found in [37, Section 2.3].

Recall that for a formula $\varphi = \wp\psi$ is easy to obtain the prefix normal form of its negation $\neg\varphi$ as $\overline{\wp}\neg\psi$, where $\overline{\wp}$ is obtained from $\wp$ by swapping every quantification from existen-

[1] The reader might observe that pushing the quantification over y outside the temporal operator does not work. Indeed, the formula $\exists y \mathrm{G}(y \wedge \mathrm{X}\neg y)$ is unsatisfiable.

tial to universal and vice-versa. From now on, by $\neg\varphi$ we denote its prenex normal form transformation.

An *alternation* in a quantification prefix $\wp$ is either a sequence $\exists x \forall y$ or a sequence $\forall x \exists y$ occurring in $\wp$. A formula of the form $\wp\psi$ is of *alternation-depth k* if $\wp$ contains exactly *k* alternations. Following the notation introduced in [18], by *k*-QLTL we denote the QLTL fragment of formulas with alternation *k*. Moreover, Σ_k^{QLTL} and Π_k^{QLTL} denote the fragments of *k*-QLTL of formulas starting with an existential and a universal quantification, respectively.

Let $\wp$ be a quantification prefix. By $\exists(\wp)$ and $\forall(\wp)$ we denote the set of variables that are quantified existentially and universally, respectively. We say that two variables x and x' belong to the same *block X* if no alternation occurs between them, i.e., they are both of the same quantification type, together with any other variable occurring in between them in $\wp$.

Note that a QLTL formula $\wp\psi$ is equivalent to any formula $\wp'\psi$ where $\wp'$ is obtained from $\wp$ by shuffling variables belonging to the same block. For this reason, it is convenient to make use of the syntactic shortcuts $\exists X\varphi \doteq \exists x_1 \ldots \exists x_k\varphi$ and $\forall X\varphi \doteq \forall x_1 \ldots \forall x_k\varphi$ with $X = \{x_1, \ldots, x_k\}$, being a block of variables in $\wp$. Formulas can then be written in the form

$$\mathrm{Qn}_1 X_1 \ldots \mathrm{Qn}_n X_n \psi$$

with $X_1, \ldots X_n$ being *maximal blocks*, meaning that every two consecutive occurrences of them are of different quantification type. More formally, it holds that $\mathrm{Qn}_i = \exists$ iff $\mathrm{Qn}_{i+1} = \forall$, for every $i < n$.

Note that also the semantics of prenex QLTL formulas can easily be lifted in terms of quantification blocks.

For a QLTL formula φ, a trace π, and an instant *i*, we obtain that

- $\pi, i \models_{\mathrm{C}} \psi$ iff $\pi, i \models_{\mathrm{LTL}} \psi$, for every quantifier-free formula ψ;
- $\pi, i \models_{\mathrm{C}} \exists X\varphi$ iff there is a trace π' that agrees with π over $-X$ such that $\pi', i \models_{\mathrm{C}} \varphi$;
- $\pi, i \models_{\mathrm{C}} \forall X\varphi$ iff for each trace π' that agrees with π over $-X$, it holds that $\pi', i \models_{\mathrm{C}} \varphi$ [2];

From now on, we might refer to variable blocks, simply as blocks. Moreover, with a slight overlap of notation, we write $X \in \exists(\wp)$ to denote that the variables of the block X are existentially quantified in $\wp$.

The satisfiability problem consists in, given a QLTL formula φ, determining whether it is satisfiable or not.

Note that every formula φ is satisfiable if, and only if, $\exists \mathrm{free}(\varphi)\varphi$ is satisfiable. This means that we can study satisfiability in QLTL for *closed* formulas, i.e., formulas where every variable is quantified.

Consider the formula $\varphi = \exists y(y \leftrightarrow \mathrm{G}x)$ with $\mathrm{free}(\varphi) = \{x\}$. This is satisfiable as, for example, the trace π obtained by combining π_x over $\{x\}$ taking always the value true with the trace π_y over $\{y\}$ assigning true at the first instant satisfies $(y \leftrightarrow \mathrm{G}x)$. Notice that $\varphi = \exists y(y \leftrightarrow \mathrm{G}x)$ is satisfiable if and only if the close formula $\exists x \exists y(y \leftrightarrow \mathrm{G}x)$ is so. Analogously, φ is valid if and only if the close formula $\forall x \exists y(y \leftrightarrow \mathrm{G}x)$ is so.

Such problem is decidable, though computationally highly intractable in general [18]. For a given natural number *k*, by *k*-EXPSPACE we denote the language of problems solved

[2] Notice that now we are dealing with variable blocks and not single variables at the time.

by a Turing machine with space bounded by $2^{2^{\cdots^{2^n}}}$, where the height of the tower is k and n is the size of the input. By convention 0-EXPSPACE denotes PSPACE.

Theorem 1 ([18]) The satisfiability problem for k-QLTL formulas is k-EXPSPACE-complete.

3 Skolem functions for QLTL semantics

We now give an alternative way to capture the semantics of QLTL, which is in terms of (second order) Skolem functions. This will allow us later to suitably restrict such Skolem functions to capture behavioral semantics, by forcing them to depend only on the past history and the current situation.

Consider two variable blocks X and Y. By $X <_\wp Y$ we denote the fact that X occurs *before* Y in $\wp$. For a given existentially quantified block $Y \in \exists(\wp)$, by $\mathrm{Dep}_\wp(Y) = \{X \in \forall(\wp) \mid X <_\wp Y\}$ we denote the blocks to which Y depends on in $\wp$. Moreover, for a given set $F \subseteq \mathrm{Var}$ of variables, sometimes referred as the *free variables block*, by $\mathrm{Dep}^F_\wp(Y) = F \cup \mathrm{Dep}_\wp(Y)$ we denote the *augmented dependency*, taking into account the additional free block. Whenever clear from the context, we omit the subscript and simply write $\mathrm{Dep}(Y)$ and $\mathrm{Dep}^F(Y)$.

The relation defined above captures the concept of *variable dependence* generated by quantifiers and free variables in a QLTL formula. Intuitively, whenever a dependence occurs between two blocks X and Y, this means that the existential choices of Y are determined by a function whose domain is given by all possible choices available for X, be it universally quantified or free in the corresponding formula. This dependence is know in first-order logic as *Skolem function* and can be described in QLTL as follows.

Definition 1 (Skolem function). For a given quantification prefix $\wp$ defined over a set $\mathrm{Var}(\wp) \subseteq \mathrm{Var}$ of variables, and a free block $F = \mathrm{Var} \setminus \mathrm{Var}(\wp)$, a function

$$\theta : (2^{F \cup \forall(\wp)})^\omega \to (2^{\exists(\wp)})^\omega$$

is called a Skolem function over $(\wp, F)$ if, for all traces $\pi_1, \pi_2 \in (2^{F \cup \forall(\wp)})^\omega$ over $\mathrm{F} \cup \forall(\wp)$ and for all blocks $Y \in \exists(\wp)$, it holds that

$$\mathrm{Prj}(\pi_1, \mathrm{Dep}^F(Y)) = \mathrm{Prj}(\pi_2, \mathrm{Dep}^F(Y)) \text{ implies } \mathrm{Prj}(\theta(\pi_1), Y) = \mathrm{Prj}(\theta(\pi_2), Y).$$

In other words, whenever π_1 and π_2 are equal over the variables to which block Y depends on, $\theta(\pi_1)$ and $\theta(\pi_2)$ are equal over the block Y.
Intuitively, a Skolem function takes traces of the free variables and (the blocks of) universally quantified variables and returns traces of (the blocks of) existentially quantified variables so that they depend only on the free variables and the universal variables that appear before them in the quantification prefix $\wp$.

Skolem functions can be used to give an alternative characterization of the semantics of QLTL formulas in prenex normal form. Given a trace π over $F \cup \forall(\wp)$, sometimes we denote

the combined trace $\hat{\theta}(\pi) \doteq \pi \uplus \theta(\pi)$, as if $\hat{\theta}$ *combines* the inputs and outputs outcomes of θ together.

Definition 2 (Skolem semantics). A QLTL formula $\varphi = \wp\psi$ is *Skolem true* over a trace π at an instant i, written $\pi, i \models_{\mathrm{S}} \varphi$, if there exists a Skolem function θ over $(\wp, \mathrm{free}(\varphi))$ such that $\hat{\theta}(\pi \uplus \pi_{\forall(\wp)}), i \models_{\mathrm{LTL}} \psi$, for every possible trace $\pi_{\forall\wp}$.

Intuitively, the Skolem semantics characterizes the truth of a QLTL formula with the existence of a Skolem function that returns the traces of the existential quantifications as function of the variables to which they depend in the formula φ.

The following theorem shows, the Skolem semantics is equivalent to the classic one. Therefore, for every formula φ and every trace π, it holds that $\pi \models_{\mathrm{S}} \varphi$ if, and only if, $\pi \not\models_{\mathrm{S}} \neg\varphi$.

Theorem 2 For every QLTL formula $\wp\psi$ and a trace $\pi_F \in (2^F)^\omega$ over the free variables block $F = \mathrm{free}(\varphi)$ of φ, it holds that

$\pi_F \models_{\mathrm{C}} \varphi$ if, and only if, $\pi_F \models_{\mathrm{S}} \varphi$.

Proof The proof proceeds by induction on the length of $\wp$. For the case of $|\wp| = 0$ it holds that $\wp = \epsilon$ is the empty sequence. This means that $\varphi = \psi$ is variable free and the classic and Skolem semantics coincide with the LTL semantics. Therefore we obtain $\pi_F \models_{\mathrm{C}} \psi$ iff $\pi_F \models_{\mathrm{S}} \psi$.

For the case of $|\wp| > 0$ we prove the two implications separately.
From the left to right direction, assume that $\pi_F \models_{\mathrm{C}} \varphi$ and distinguish two cases:

- $\wp = \exists X\wp'$. Thus, there exists a trace $\pi_X \in (2^X)^\omega$ such that $\pi_F \uplus \pi_X \models_{\mathrm{C}} \wp'\psi$. By induction hypothesis, we have that $\pi_F \uplus \pi_X \models_{\mathrm{S}} \wp'\psi$ and so that there exists a Skolem function θ' over $(\wp', F \cup \{X\})$ such that $\hat{\theta}'(\pi_F \uplus \pi_X \uplus \pi') \models_{\mathrm{LTL}} \psi$, for every $\pi' \in (2^{\forall(\wp')})^\omega$. Now, consider the Skolem function θ over $(\wp, F)$ defined as $\theta(\pi_F \uplus \pi') = \theta'(\pi_F \uplus \pi_X \uplus \pi'_{-X}) \uplus \pi_X$ for every $\pi' \in (2^{\forall(\wp)})^\omega$. This implies that $\hat{\theta}(\pi_F \uplus \pi') \models_{\mathrm{LTL}} \psi$ for every $\pi' \in (2^{\forall(\wp)})^\omega$, and so that $\pi_F \models_{\mathrm{S}} \varphi$.
- $\wp = \forall X\wp'$. Then, it holds that $\pi_F \uplus \pi_X \models_{\mathrm{C}} \wp'\psi$ for every $\pi_X \in (2^X)^\omega$. By induction hypothesis, for every $\pi_X \in (2^X)^\omega$ there exists a Skolem function $\theta\pi_X$ over $(\wp', F \cup \{X\})$ such that $\hat{\theta}\pi_X(\pi_F \uplus \pi_X \uplus \pi') \models_{\mathrm{LTL}} \psi$ for every $\pi' \in (2^{\forall(\wp')})^\omega$. Now, consider the Skolem function θ over $(\wp, F)$ defined as $\theta(\pi_F \uplus \pi') = \theta_{\pi'_X}(\pi_F \uplus \pi'_X \uplus \pi'_{-X})$. It holds that $\hat{\theta}(\pi_F \uplus \pi') \models_{\mathrm{LTL}} \psi$ for every $\pi' \in (2^\wp)^\omega$, which means that $\pi_F \models_{\mathrm{S}} \varphi$.

For the right to left direction, assume that $\pi_F \models_{\mathrm{S}} \wp\psi$. Then, there exists a Skolem function θ over $(\wp, F)$ such that $\hat{\theta}(\pi_F \uplus \pi) \models_{\mathrm{LTL}} \psi$ for every $\pi \in (2^{\forall(\wp)})^\omega$. Here, we also distinguish the two cases.

- $\wp = \exists X\wp'$. Observe that $\mathrm{Dep}^F(X) = F$. Then it holds that $\theta(\pi_F \uplus \pi)(X) = \theta(\pi_F \cup \pi')(X) = \pi_X$ for

every $\pi, \pi' \in (2^{\forall(\wp')})^\omega$. Now, define the Skolem function θ' over $(\wp', F \cup \{X\})$ as $\theta'(\pi_F \uplus \pi_X \cup \pi') = \mathrm{Prj}(\theta(\pi_F \uplus \pi_X \uplus \pi'), -X)$ outputting the same as θ except for the trace of the block variable X. It holds that $\hat{\theta}'(\pi_F \uplus \pi_X \uplus \pi') \models_{\mathrm{LTL}} \psi$ for each $\pi' \in (2^{\forall(\wp')})\omega$ and so, by induction hypothesis, that $\pi_F \uplus \pi_X \models_{\mathrm{C}} \wp'\psi$, which in turns implies that $\pi_F \models_{\mathrm{C}} \exists X \wp'\psi$ and so that $\pi_F \models_{\mathrm{C}} \varphi$.

- $\wp = \forall X \wp'$. Observe that $\forall(\wp) = \forall(\wp') \cup \{X\}$, and so that θ is also a Skolem function over $(\wp', F \cup \{X\})$. This implies that, for each $\pi_X \in (2^X)^\omega$, it holds that $\hat{\theta}(\pi_F \uplus \pi_X \uplus \pi') \models_{\mathrm{LTL}} \psi$ for every $\pi' \in (2^{\forall(\wp')})^\omega$. By induction hypothesis, we obtain that, for every $\pi_X \in (2^{\mathrm{X}})^\omega$, it holds that $\pi_F \uplus \pi_X \models_{\mathrm{C}} \wp'\psi$, which in turns implies that $\pi_F \models_{\mathrm{C}} \forall X \wp'\psi$ and so that $\pi_F \models_{\mathrm{C}} \varphi$. □

4 Behavioral QLTL (QLTL$_B$)

The classic semantics of QLTL requires to consider at once the evaluation of the variables on the whole trace. This gives rise to counter-intuitive phenomena. Consider the formula $\forall x \exists y(\mathrm{G}x \leftrightarrow y)$. Such a formula is satisfiable. Indeed, on the one hand, for the trace assigning always true to x, the trace that makes y true at the beginning satisfies the temporal part. On the other hand, for every other trace making x false sometimes, the trace that makes y false at the beginning satisfies the temporal part. However, in order to correctly interpret y on the first instant, one needs to know in advance the entire trace of x. Such requirement is practically impossible to fulfill and does not reflect the notion of *reactive systems*, where the agent variables at the k-th instant of the computation depend only on the past assignments of the environment variables. Such principle is often referred to as *behavioral* in the context of strategic reasoning, see e.g., [21, 24].

Here, we introduce an alternative semantics for QLTL, which is based on the idea that the *existential variables are controlled by the agent* and the *universally quantified variables are controlled by the environment*. We require such control functions to be processes in the sense of [1], i.e., the next move depends only on the past history and the present, but not the future. Moreover the choices of the existential variables can depend only on the universal variables coming earlier in the quantification prefix. In other words this semantics allows for *partial observability* of the uncontrollable variables (i.e., the universally quantified variables).

To formally define the semantics, we suitably constrain Skolem functions to make them behavioral, i.e., processes.

Specifically we introduce *behavioral* QLTL, denoted QLTL$_{\mathrm{B}}$, a logic with the same syntax as of prenex normal form QLTL, namely:

$$\varphi ::= \exists x \varphi \mid \forall x \varphi \mid \psi$$

where ψ is an LTL formula over the the propositional variables Var. However, while the syntax is the same of QLTL, the semantics of QLTL$_{\mathrm{B}}$ is defined in terms of behavioral Skolem functions.

Definition 3 (Behavioral Skolem function). For a given quantification prefix $\wp$ defined over blocks of propositional variables in Var and a block F of free variables, a Skolem function θ over $(\wp, F)$ is *behavioral* if, for all $\pi_1, \pi_2 \in (2^{F \cup \forall(\wp)})^\omega$, $k \in \mathbb{N}$, and $Y \in \exists(\wp)$, it holds that

$$\begin{gathered}\mathrm{Prj}(\pi_1(0,k), \mathrm{Dep}^F(Y)) = \mathrm{Prj}(\pi_2(0,k), \mathrm{Dep}^F(Y)) \\ \text{implies} \\ \mathrm{Prj}(\theta(\pi_1)(0,k), Y) = \mathrm{Prj}(\theta(\pi_2)(0,k), Y).\end{gathered}$$

The behavioral Skolem functions capture the fact that the trace of existentially quantified variables depends only on the past and present values of free and universally quantified variables. This offers a way to formalize the semantics of $\mathrm{QLTL_B}$ as follows.

Definition 4 A $\mathrm{QLTL_B}$ formula $\varphi = \wp\psi$ is true over a trace π in an instant i, written $\pi, i \models_{\mathrm{B}} \wp\psi$, if there exists a behavioral Skolem function θ over $(\wp, \mathrm{free}(\varphi))$ such that $\hat{\theta}(\pi \uplus \pi'), i \models_{\mathrm{C}} \psi$ for every $\pi' \in (2^{\mathrm{free}(\varphi) \cup \forall(\wp)})^\omega$.

A $\mathrm{QLTL_B}$ formula φ is true on a trace π, written $\pi \models_{\mathrm{B}} \varphi$, if $\pi, 0 \models_{\mathrm{B}} \varphi$. A formula φ is *satisfiable* if it is true on some trace and *valid* if it is true in every trace. Consider again the formula $\varphi = \exists y(y \leftrightarrow \mathrm{G}x)$ with $\mathrm{free}(\varphi) = \{x\}$, now in $\mathrm{QLTL_B}$. This is satisfiable again. Indeed, consider the behavioral Skolem function θ such that $\theta(\pi_x)(0,0) = \mathrm{true}$ and $\theta(\pi_x)(0,k) = \mathrm{false}$ for each $k > 0$. Now, for the trace π obtained by combining π_x over $\{x\}$ taking always the value true with the trace $\pi_y = \theta(\pi_x)$ over $\{y\}$ generated by the Skolem function θ, we have that π satisfies $(y \leftrightarrow \mathrm{G}x)$.

Again, notice that $\varphi = \exists y(y \leftrightarrow \mathrm{G}x)$ is satisfiable if and only if the close formula $\exists x \exists y(y \leftrightarrow \mathrm{G}x)$ is so. Indeed, now notice that the Skolem function chose both the values of x and y as needed in $(y \leftrightarrow \mathrm{G}x)$. However, the formula φ is not valid. Indeed, the closed formula $\forall x \exists y(y \leftrightarrow \mathrm{G}x)$ is neither satisfiable nor valid in $\mathrm{QLTL_B}$ since, in order to set the value of y appropriately, one should be able to observe the whole trace π_x and, since behavioral Skolem functions depend only on history, this cannot be done.

Observe that also the negation of $\forall x \exists y(y \leftrightarrow \mathrm{G}x)$ is not satisfiable. Indeed, the formula $\exists y \forall x(x \not\leftrightarrow \mathrm{G}y)$ cannot have a Skolem function that sets the values of y appropriately without seeing x at the first instant. This is a common phenomenon, as it also happens when considering the behavioral semantics of logic for the strategic reasoning [21, 24].

Consider instead the formula $\varphi = \exists y \mathrm{G}(y \leftrightarrow x)$. This is both satisfiable and valid. Indeed, in the case of satisfiability, the closed formula $\exists x \exists y \mathrm{G}(y \leftrightarrow x)$ is satisfiable as the behavioral Skolem function can chose the values of x and y appropriately. For the case of validity, the closed formula $\forall x \exists y \mathrm{G}(y \leftrightarrow x)$ is satisfiable, as the Skolem function can set the value of y in dependence of the history of values for x (in particular, the last one) in a suitable way. Instead the formula $\exists y \forall x \mathrm{G}(y \leftrightarrow x)$ is not satisfiable (neither valid) since the Skolem function needs to chose the values for y independently (i.e., without observing) the values of x.

5 Capturing advanced forms of Planning in QLTL$_B$

In order to gain some intuition on QLTL$_B$, it is interesting to see how QLTL$_B$ can capture advanced forms of Planning. We assume some familiarity with Planning in AI, see [27, 38]. In planning, we typically have a: *domain D* (here including the initial state) describing the dynamics of the environment, i.e., what happens when the agent performs its actions; a *goal G* that the agent has to accomplish in the domain. The various forms of planning can be seen as a game between the agent controlling the *actions* and environment controlling the *fluents*. Given an agent's action, the environment responds by setting the fluents according to the specification in *D*. The agent has to come up with actions that eventually enforce the goal *G*. Typically the goal is reaching a state with certain properties (values of fluents) but here we consider temporally extended goals, so the goal is a specification of desirable traces rather than states [4].

Here we consider several forms of planning where the fluents to the agent are: (i) totally invisible (*conformant planning*); (ii) totally visible–but not controllable (*contingent planning with full observability*); (iii) or partially visible (*contingent planning with partial observability*).

In the following, we assume to have a LTL formula φ_D that captures the domain *D* (including the initial state), and another LTL formula φ_g that captures the agent goal *G*. Such formulas are on fluents, controlled by the environment, for which we use the variables *X* possibly with subscripts, and actions, controlled by the agent, for which we used the variable *Y* possibly with subscripts.

Notice that by using LTL to express the domain we can actually capture not only standard Markovian domains, but also non-Markovian ones in which the reaction of the environment depends on the whole history, as well as, liveness constrains on the environment dynamics. So φ_D can be seen as denoting the set of traces that satisfy the (temporally extended) domain specifications *D*.

The general formula for a planning problem is of the form:

$$\varphi = \varphi_D \rightarrow \varphi_g$$

which says that on the infinite runs where the environment acts as prescribed by φ_D the goal φ_g holds [11, 14]. Note that φ does not mention strategies but only traces, so it is not very useful in isolation to solve planning, i.e., to show the existence of a plan/strategy that guarantees φ independently of the environment's behavior. To capture this, we are going to use second order quantification of QLTL$_B$.

In all the formulas below, the blocks *X* are the fluents and blocks *Y* are the actions (coded in binary for simplicity).

Consider the QLTL$_B$ formula:

$$\exists Y \forall X \varphi$$

this is looking for an assignment of the actions *Y* such that for every assignment of the fluents *X* the resulting LTL formula φ holds. This formula captures *conformant planning* [7]. Note that the values of *Y*, i.e., the choice of actions at each point in time, do not depend on *X*. That is the plan (the Skolem function deciding *Y*) does not see the evolution of the fluents *X*.

This is the reason why the plan is conformant. Note also that in this case the fact that X are assigned through a behavioral Skolem function or any Skolem function is irrelevant, since we do not see the values of X anyway when choosing the Skolem function for Y (i.e., the plan). So this form of planning could be captured through standard QLTL as well.

Consider the QLTL_B formula:

$$\forall X \exists Y \varphi$$

this states that at every point in time for every value of the fluents X there exists an action Y such that the resulting trace satisfies φ. This captures *contingent planning with full observability*, i.e., (strong) planning in *Fully Observable Nondeterministic Domains (FOND)* [26, 27]. Here the fact that Y at the current instant may depend only on the past and current values of X of the behavioral semantics is critical. Otherwise the choices of action Y would depend on the future values of fluents X, that is, the plan would **not** be a process but would forecast the future, which is usually impossible in practice. Note that with QLTL_B formulas of the form $\forall X \exists Y \psi$, where ψ is an arbitrary LTL formula, we capture LTL synthesis (for realizing the LTL specification ψ) [12].

Now consider the QLTL_B formula

$$\forall X_1 \exists Y \forall X_2 \varphi.$$

It is similar to the previous one but now we have split the fluents X into X_1 and X_2 and the actions Y are allowed to depend on X_1 but not on X_2. In other words, the Skolem function for Y may depend on the previous and current values of X_1 but does **not** depend on the values of X_2. This captures *contingent planning with partial observability*, i.e., (strong) planning in *Partially Observable Nondeterministic Domains (POND)*, where some fluents are observable (X_1) and some are not (X_2), and indeed the plan can only depend on the observable ones [27, 39]. Note that with QLTL_B formulas of the from $\forall X_1 \exists Y \forall X_2 \psi$, where ψ is an arbitrary LTL formula, we capture synthesis under incomplete information (for realizing the LTL specification ψ) [31].

Notice also that we can indeed include fairness assumptions in φ_D and hence in φ, so with some care, see [40], the above two QLTL_B formulas can capture also strong cyclic plans [8, 27].

As we allow more quantifier nesting we get more and more sophisticated forms of planning. For example the QLTL_B formula:

$$\forall X_1 \exists Y_1 (\ldots) \forall X_n \exists Y_n \varphi$$

captures a centralized planning for multiple plan actuators with hierarchically reduced partial observability, with the innermost plan actuator, controlling Y_n, solving a FOND planning instance. Similarly

$$\forall X_1 \exists Y_1 (\ldots) \forall X_n \exists Y_n \forall X_{n+1} \varphi$$

captures a centralized planning for multiple plan actuators with hierarchically reduced partial observability, with the innermost plan actuator, controlling Y_n, solving a POND planning instance. Instead

$$\exists Y_1 \forall X_1(\ldots)\exists Y_{n-1}\forall X_{n-1}\exists Y_n \varphi$$

captures a centralized planning for multiple plan actuators with hierarchically reduced partial observability, with the outermost actuator, controlling Y_1, solving a conformant planning instance and the innermost, controlling Y_n, solving a FOND planning instance. Similarly

$$\exists Y_1 \forall X_1(\ldots)\exists Y_{n-1}\forall X_{n-1}\exists Y_n \forall X_n \varphi$$

captures a centralized planning for multiple plan actuators with hierarchically reduced partial observability, with the outermost actuator, controlling Y_1, solving a conformant planning instance and the innermost, controlling Y_n, solving a POND planning instance.

Note that, these last forms of planning have never been studied in detail in the AI literature. However the corresponding form of synthesis has indeed been investigated under the name of *distributed synthesis* [28, 30]. Distributed synthesis concerns the coordination of a number of agents, each with partial observability on the environment and on the other agents, so as to enforce together an LTL formula. Several visibility architectures among agents have been considered, including those that allow for *information forks*, that is, situations in which two agents receive information from the environment in a way that they cannot completely deduce the information received by the other agent. In general distributed synthesis is undecidable [28]. However, it has been proven that the absence of information forks is sufficient to guarantee the decidability of synthesis [30]. Specifically, without information forks it is possible to arrange the agents in a sort of information hierarchy, which leads to decidability [30]. Incidentally, this is the form of uniform distributed synthesis that is captured by the above QLTL_B formulas. Indeed we will show later that solving a distributed synthesis with hierarchical information architectures can be done optimally by reduction to QLTL_B satisfiability of the formulas presented above.

6 QLTL_B properties

Clearly, since QLTL_B shares the syntax with QLTL, all the definitions that involve syntactic elements, such as free variables and alternation, apply to this variant the same way.

As for QLTL, the satisfiability of a QLTL_B formula φ is equivalent to the one of $\exists\text{free}(\varphi)\varphi$, as well as the validity is equivalent to the one of $\forall\text{free}(\varphi)\varphi$. However, the proof is not as straightforward as for the classic semantics case.

Theorem 3 For every QLTL_B formula $\varphi = \wp\psi$, φ is satisfiable if, and only if, $\exists\text{free}(\varphi)\varphi$ is satisfiable. Moreover, φ is valid if, and only if, $\forall\text{free}(\varphi)\varphi$ is valid.

Proof We show the proof only for satisfiability, as the one for validity is similar. The proof proceeds by double implication.

From left to right, assume that φ is satisfiable, therefore there exists a trace π over $F = \text{free}(\varphi)$ such that $\pi \models_{\text{B}} \varphi$, which in turns implies that there exists a behavioral Skolem function θ over $(\wp, F)$ such that $\hat{\theta}(\pi \Cup \pi') \models_{\text{C}} \psi$ for every trace $\pi' \in (2^{\forall(\wp)})^\omega$. Consider the function $\theta' : (2^{\forall(\wp)})^\omega \to (2^{\exists(\wp)\cup F})^\omega$ defined as $\theta'(\pi') = \theta(\pi \Cup \pi') \Cup \pi$, for every $\pi' \in (2^{\forall(\wp)})^\omega$. Clearly, it is a behavioral Skolem function over $(\exists F\wp, \emptyset)$ such that $\hat{\theta}'(\pi') \models \psi$ for every $\pi' \in (2^{\forall(\wp)})^\omega$, which implies that $\exists F\varphi$ is satisfiable.

From right to left,

we have that $\exists F\varphi$ is satisfiable, which means that there exists a behavioral Skolem function θ over $(\exists F\varphi, \emptyset)$ such that $\hat{\theta}(\pi) \models_{\text{LTL}} \psi$ for every $\pi \in (2^{\forall(\wp)\cup\{F\}})^\omega$. Observe that $\text{Dep}^F_{\exists F\wp}(F) = \emptyset$ [3], and so that $\theta(\pi)(F) = \theta(\pi')(F) = \pi_F$ for every $\pi, \pi' \in (2^{\forall(\wp)})^\omega$. Thus, consider the behavioral Skolem function θ' over $(\wp, F)$ defined as $\theta'(\pi'_F \Cup \pi) = \theta(\pi_F \Cup \pi)$, for every $\pi'_F \in (2^F)^\omega$ and $\pi \in (2^{\forall(\wp)})^\omega$, from which it follows that $\theta'(\pi_F \cup \pi) \models_{\text{LTL}} \psi$ for every $\pi \in (2^{\forall(\wp)})^\omega$, from which we derive that $\pi_F \models_{\text{B}} \wp\psi$, and so that φ is satisfiable. □

Note that every behavioral Skolem function is also a Skolem function. This means that a formula φ interpreted as QLTL_{B} is true on π implies that the same formula is true on π also when it is interpreted as QLTL. The reverse, however, is not true, as we have seen this when discussing the satisfiability of the formula $\varphi = \forall x \exists y(y \leftrightarrow \text{G}x)$. Indeed, we have.

Lemma 1 *For every QLTL_B formula φ and a trace π over the set $\text{free}(\varphi)$ of free variables, if $\pi \models_B \varphi$ then $\pi \models_C \varphi$. On the other hand, there exists a formula φ and a trace π such that $\pi \models_C \varphi$ but not $\pi \models_B \varphi$.*

Proof The first part of the theorem follows from the fact that every behavioral Skolem function is also a Skolem function and so, if $\pi \models_{\text{B}} \varphi$, clearly also $\pi \models_{\text{S}} \varphi$ and so, from Theorem 2, that $\pi \models_{\text{C}} \varphi$.

For the second part, consider the formula $\varphi = \forall x \exists y(\text{G}x \leftrightarrow y)$. We have already shown that such formula is satisfiable. However, it is not behavioral satisfiable. Indeed, assume by contradiction that it is behavioral satisfiable and let θ be a behavioral Skolem function such that $\theta \models (\text{G}x \leftrightarrow y)$ for every trace $\pi \in (2^x)^\omega$. Now consider two traces π_1 over x that always assigns true, and π_2 that assigns true on x at the first iteration and then always false. It holds that $\pi_1(0) = \pi_2(0)$ and therefore, since $x \in \text{Dep}(y)$ and θ is behavioral, it must be the case that $\text{Prj}(\theta(\pi_1)(0), y) = \text{Prj}(\theta(\pi_2)(0), y)$. Now, if such value is $\text{Prj}(\theta(\pi_1)(0), y) = \text{false}$, then it holds that $\hat{\theta}(\pi_1) \not\models_{\text{C}} (\text{G}x \leftrightarrow y)$. On the other hand, if $\text{Prj}(\theta(\pi_2)(0), y) = \text{true}$, then it holds that $\hat{\theta}(\pi_2) \not\models_{\text{C}} (\text{G}x \leftrightarrow y)$, which means that $\theta \not\models_{\text{C}} (\text{G}x \leftrightarrow y)$, a contradiction. □

Lemma 1 has implications also on the meaning of negation in QLTL_{B}. Indeed, both the formula $\varphi = \forall x \exists y(\text{G}x \leftrightarrow y)$ and its negation are not satisfiable, that is $\not\models_{\text{B}} \forall x \exists y(\text{G}x \leftrightarrow y)$ and $\not\models_{\text{B}} \exists x \forall y \neg(\text{G}x \leftrightarrow y)$.

This is a common phenomenon, as it also happens when considering the behavioral semantics of logic for the strategic reasoning [21, 24].

[3] Note that the formula is a sentence, i.e., there are no free variables.

7 QLTL$_B$ satisfiability

There are three syntactic fragments for which QLTL and QLTL$_\mathrm{B}$ are equivalent. Precisely, the fragments $\Pi_0^{\mathrm{QLTL_B}}$, $\Sigma_0^{\mathrm{QLTL_B}}$, and $\Sigma_1^{\mathrm{QLTL_B}}$. Recall that $\Pi_0^{\mathrm{QLTL_B}}$ formulas are of the form $\forall X \varphi_{\mathrm{LTL}}$, whereas $\Sigma_0^{\mathrm{QLTL_B}}$ formulas are of the form $\exists Y \varphi_{\mathrm{LTL}}$. Finally, $\Sigma_1^{\mathrm{QLTL_B}}$ formulas are of the form $\exists Y \forall X \varphi_{\mathrm{LTL}}$. The reason is that the sets of Skolem and behavioral Skolem functions for these formulas coincide, and so the existence of one implies the existence of the other.

Theorem 4 For every QLTL$_\mathrm{B}$ formula $\varphi = \wp\psi$ in the fragments $\Pi_0^{\mathrm{QLTL_B}}$, $\Sigma_0^{\mathrm{QLTL_B}}$, and $\Sigma_1^{\mathrm{QLTL_B}}$ and every trace π, it holds that $\pi \models_\mathrm{B} \varphi$ if, and only if, $\pi \models_\mathrm{C} \varphi$.

Proof The proof proceeds by double implication. From left to right, it follows from Lemma 1. From right to left, consider first the case that $\varphi \in \Pi_0^{\mathrm{QLTL}}$. Observe that $\exists(\wp) = \emptyset$ and so the only possible Skolem function θ returns empty on every possible trace $\pi \uplus \pi' \in (2^{\mathrm{free}(\varphi) \cup \forall(\wp)})^\omega$. Such Skolem function is trivially behavioral and so we have that $\pi \models_\mathrm{S} \varphi$ implies $\pi \models_\mathrm{B} \varphi$.

For the case of $\varphi \in \Sigma_0^{\mathrm{QLTL}} \cup \Sigma_1^{\mathrm{QLTL}}$, assume that $\pi, \models_\mathrm{S} \varphi$ and let θ be a Skolem function such that $\hat{\theta}(\pi \cup \pi') \models_{\mathrm{LTL}} \psi$ for every $\pi' \in (2^{\forall(\wp)})^\omega$. Observe that, for every $Y \in \exists(\wp)$, it holds that $\mathrm{Dep}_\wp = \emptyset$ and so the values of Y depend only on the free variables in φ. This means that θ is necessarily behavioral, as such condition is vacuously satisfied. This implies that $\pi \models_\mathrm{B} \varphi$. □

Theorem 4 shows that for these three fragments of QLTL$_\mathrm{B}$, satisfiability can be solved by employing QLTL satisfiability. This also comes with the same complexity, as we just interpret the QLTL$_\mathrm{B}$ formula directly as QLTL one.

Corollary 1 *Satisfiability for the fragments $\Pi_0^{QLTL_B}$ and $\Sigma_0^{QLTL_B}$ is PSPACE-complete. Moreover, satisfiability for the fragment $\Sigma_1^{QLTL_B}$ is EXPSPACE-complete.*

We now turn into solving satisfiability for QLTL$_\mathrm{B}$ formulas that are not in fragments $\Pi_0^{\mathrm{QLTL_B}}$, $\Sigma_0^{\mathrm{QLTL_B}}$, and $\Sigma_1^{\mathrm{QLTL_B}}$. Analogously to the case of QLTL, note that Theorem 3 allows to restrict our attention to closed formulas. We use an automata-theoretic approach inspired by the one employed in the synthesis of distributed systems [29, 30]. This requires some definitions and results, presented below.

For a given set Υ of directions, a Υ-*tree* is a set $T \subseteq \Upsilon^*$ of finite words. The elements of T are called nodes, and the empty word ε is called *root*. The Υ-tree $T = \Upsilon^*$ is called *full*. For every $x \in T$, the nodes $x \cdot c \in T$ are called children. We say that $c = \mathrm{dir}(x \cdot c)$ is the direction of the node $x \cdot c$, and we fix some $\mathrm{dir}(\varepsilon) = c_0 \in \Upsilon$ to be the direction of the root.

The following definitions are given only on full trees, as they are the only kind we deal with in the rest of the paper. Given two finite sets Υ and Σ, a Σ-labeled Υ-tree is a pair $\langle \Upsilon^*, l \rangle$ where $l : \Upsilon^* \to \Sigma$ maps/labels every node of Υ^* into a letter in Σ.

For a set $\Theta \times \Upsilon$ of directions and a node $x \in (\Theta \times \Upsilon)^*$, $\mathrm{hide}_\Upsilon(x)$ denotes the node in Θ^* obtained from x by replacing (ϑ, υ) with ϑ in each letter of x.

An *alternating automaton* $\mathcal{A} = (\Sigma, Q, q_0, \delta, \alpha)$ runs over full Σ-labeled Υ-trees, for a predefined set of directions Υ. The set of states Q is finite with q_0 being a designated initial

state, while $\delta : Q \times \Sigma \to \mathcal{B}^+(Q \times \Upsilon)$ denotes a transition function, returning a positive Boolean formula over pairs of states and directions, and α is an acceptance condition.

We say that $\mathcal{A}$ is *nondeterministic*, and denote it with the symbol $\mathcal{N}$, if every transition returns a positive Boolean formula with only disjunctions. Moreover, that it is *deterministic*, and denote it with the symbol $\mathcal{D}$, if every transition returns a single state.

Note that the transition relation δ of a nondeterministic automaton can be seen as a function $\delta : Q \times \Sigma \to 2^{\Upsilon \to Q}$, mapping every pair (q, σ) of state and alphabet symbol to a set of functions $f \in \delta(q, \sigma)$ each of them mapping a direction to a state.

A run tree of $\mathcal{A}$ on a Σ-labeled Υ tree $\langle \Upsilon^*, l \rangle$ is a $Q \times \Upsilon$-labeled tree where the root is labeled with $(q_0, l(\varepsilon))$ and where, for a node x with a label (q, x), and a set of children $\mathrm{child}(x)$, the labels of these children have the following properties:

- for all $y \in \mathrm{child}(x)$, the label of y is of the form $(q_y, x \cdot c_y)$ such that (q_y, c_y) is an atom of the formula $\delta(q, l(x))$ and
- the set of atoms defined by the children of x satisfies $\delta(q, l(x))$.

We say that α is a *parity* condition if it is a function $\alpha : Q \to C(\subset \mathbb{N})$ mapping every state to a natural number, sometimes referred as color. Alternatively, it is a *Streett* condition if it is a set of pairs $\{(G_i, R_i)\}_{i \in I}$, where each G_i, R_i is a subset of Q. An infinite path ρ over Q fulfills a parity condition α if the highest color mapped by α over ρ that appears infinitely often is even. The path ρ fulfills a Streett condition if, for every $i \in I$, either an element of G_i or no element of R_i occurs infinitely often on ρ. A run tree is *accepting* if all its paths fulfill the acceptance condition α. A tree is accepted by $\mathcal{A}$ if there is an accepting run tree over it. By $\mathcal{L}(\mathcal{A})$ we denote the set of trees accepted by $\mathcal{A}$. An automaton $\mathcal{A}$ is *empty* if $\mathcal{L}(\mathcal{A}) = \emptyset$.

For a Σ-labeled Υ-tree $\langle \Upsilon^*, l_\Sigma \rangle$ and a Ξ-labeled $\Upsilon \times \Theta$-tree $\langle (\Upsilon \times \Theta)^*, l_\Xi \rangle$, their *composition*, denoted $\langle \Upsilon^*, l_\Sigma \rangle \oplus \langle (\Upsilon \times \Theta)^*, l_\Xi \rangle$ is the $\Xi \times \Sigma$-labeled $\Upsilon \times \Theta$-tree $\langle (\Upsilon \times \Theta)^*, l \rangle$ such that, for every $x \in (\Upsilon \times \Theta)^*$, it holds that $l(x) = l_\Xi(x) \cup l_\Sigma(\mathrm{hide}_\Theta(x))$.

Observe that the Υ-component appears in both the trees. Their composition, indeed, can be seen as an extension of the labeling l_Ξ with the labeling l_Σ in a way that the choices for it are oblivious to the Θ-component of the direction. A more general definition of tree composition is given in [30].

For a set $\mathcal{T}$ of $\Xi \times \Sigma$-labeled $\Upsilon \times \Theta$-trees, $\mathrm{shape}_{\Xi,\Upsilon}(\mathcal{T})$ is the set of Σ-labeled Υ-trees $\langle \Upsilon^*, l_\Sigma \rangle$ for which there exists a Ξ-labeled $\Upsilon \times \Theta$-tree $\langle (\Upsilon \times \Theta)^*, l_\Xi \rangle$ such that $\langle \Upsilon^*, l_\Sigma \rangle \oplus \langle (\Upsilon \times \Theta)^*, l_\Xi \rangle \in \mathcal{T}$. Intuitively, the shape operation performs a nondeterministic guess on the Σ-component of the trees by taking into account only the Υ-component of the directions. This allows to refine the set of trees into those for which a decomposition consistent with this limited dependence is possible. Interestingly, being this nondeterministic guess similar to an existential projection, we can also refine a (nondeterministic) parity tree automaton $\mathcal{N}$ in order to recognize the shape language of $\mathcal{L}(\mathcal{N})$. Indeed, consider a nondeterministic parity tree automaton $\mathcal{N} = (\Xi \times \Sigma, Q, q_0, \delta, \alpha)$ recognizing $\Xi \times \Sigma$-labeled Υ-trees, we define the alternating automaton $\mathrm{change}_{\Xi,\Upsilon}(\mathcal{N}) = (\Sigma, Q, q_0, \delta', \alpha)$ that recognizes Σ-labeled Υ-trees where

$$\delta'(q, \sigma) = \bigvee_{\xi \in \Xi, f \in \delta(q,(\xi,\sigma))} \bigwedge_{\upsilon \in \Upsilon} (f(\upsilon), (\xi, \sigma)).$$

The alternating automaton $\mathrm{change}_{\Xi,\Upsilon}(\mathcal{N})$ projects out the the Ξ-labels by embedding them in nondeterministic guesses in its transition. As a result, it accepts a (Σ)-labeled Υ-tree iff there exists a $(\Xi \times \Sigma)$-labeled Υ-tree accepted by the original nondeterministic automaton $\mathcal{N}$.

Finkbeiner and Schewe [30] characterized the language of $\mathrm{change}_{\Xi,\Upsilon}(\mathcal{N})$ in terms of the shape operation over the trees accepted by the original nondeterministic automaton $\mathcal{N}$:

Theorem 5 ([30, Theorem 4.11]) For every nondeterministic parity tree automaton $\mathcal{N}$ over $\Xi \times \Sigma$-labeled $\Upsilon \times \Theta$-trees, it holds that $\mathcal{L}(\mathrm{change}_{\Xi,\Upsilon}(\mathcal{N})) = \mathrm{shape}_{\Xi,\Upsilon}(\mathcal{L}(\mathcal{N}))$.

We can apply the change operation only to nondeterministic automata. This means that, in order to recognize the shape language of a parity alternating automaton $\mathcal{A}$, we first need to turn it into a nondeterministic one. This can be done by means of two steps: we first turn $\mathcal{A}$ into a nondeterministic Streett automaton $\mathcal{N}_S$ that recognize the same language $\mathcal{L}(\mathcal{N}_S) = \mathcal{L}(\mathcal{A})$, and then turn it into a nondeterministic parity $\mathcal{N}$ such that $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{N}_S) = \mathcal{L}(\mathcal{A})$. If $\mathcal{A}$ has $n = |Q|$ states and $c = |C|$ colors, then the automaton $\mathcal{N}_S$ has $n^{O(c \cdot n)}$ states and $O(c \cdot n)$ pairs such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{N}_S)$ [41]. In addition, if the nondeterministic Streett automaton $\mathcal{N}_S$ has m states and p pairs, we can build a nondeterministic parity automaton $\mathcal{N}$ with $p^{O(p)} \cdot m$ states and $O(p)$ colors [30]. By applying these two constructions, we then transform an alternating parity automaton $\mathcal{A}$ into a nondeterministic one $\mathcal{N}$ accepting the same tree-language. Note that $\mathcal{N}$ is of size single exponential with respect to $\mathcal{A}$. Indeed, we obtain it with $n' = O(c \cdot n)^{O(c \cdot n)} = n^{O(c \cdot n)}$ states [4] and $c' = O(c \cdot n)$ colors. By $\mathrm{ndet}(\mathcal{A}) = \mathcal{N}$ we denote the transformation of an alternating parity automaton into a nondeterministic parity one.

From now on, we consider closed $\mathrm{QLTL_B}$ formulas being of the form

$$\wp\psi = \exists Y_1 \forall X_1 \ldots \exists Y_n \forall X_n \psi$$

with Y_1 and X_n being possibly empty. Therefore, we refer to θ as a behavioral Skolem function over $\wp$, as the set $F = \emptyset$ is always empty. Moreover, we define $\hat{X}_i = \bigcup_{j \leq i} X_j$ and $\hat{Y}_i = \bigcup_{j \leq i} Y_j$, with $X = \hat{X}_0$ and $Y = \hat{Y}_0$, respectively. Finally, we define $\check{X}_i = \bigcup_{j > i} X_j$ and $\check{Y}_i = \bigcup_{j > i} Y_j$, respectively.

A behavioral Skolem function θ over $\wp$ can be regarded as the labeling function of a 2^Y-labeled 2^X-tree. In addition, such labeling fulfills a *compositional* property, as it is expressed in the following lemma.

Lemma 2 *Let $\wp = \exists Y_1 \forall X_1 \ldots \exists Y_n \forall X_n$ be a prefix quantifier. A 2^Y-labeled 2^X-tree θ is a behavioral Skolem function over $\wp$ iff there exists a tuple $\theta_1, \ldots, \theta_n$, where θ_i is a 2^{Y_i}-labeled $2^{\hat{X}_i}$-tree, such that $\theta = \theta_1 \oplus \ldots \oplus \theta_n$.*

Proof The proof proceeds by double implication. From left to right, consider a behavioral Skolem function θ and, for every $1 \leq i \leq n$, consider the 2^{Y_i}-labeled $2^{\hat{X}_i}$-tree θ_i, defined as $\theta_i(x) = \mathrm{Prj}(\theta(x \times x'), Y_i)$ where $x \in (2^{\hat{X}_i})^*$ and $x' \in (2^{X \setminus \hat{X}_i})^*$. Note that

[4] The last equivalence holds because the number c of colors is bounded by the number n of states.

$\mathrm{Dep}_{\wp}(Y_i) = \hat{X}_i$ and so the definition of θ_i over x does not really depend on the values in x', therefore it is well-defined. By applying the definition of tree composition, it easily follows that $\theta = \theta_1 \oplus \ldots \oplus \theta_n$.

For the right to left direction, let $\theta_1, \ldots, \theta_n$ be labeled trees and consider the composition $\theta = \theta_1 \oplus \ldots \oplus \theta_n$. From the definition of tree composition, it follows that for every i, $\mathrm{Prj}(\theta(x), Y_i) = \theta_i(x_{\hat{X}_i})$, which fulfills the requirement for θ of being a behavioral Skolem function over $\wp$. □

We now show how to solve satisfiability for $\mathrm{QLTL_B}$ with an automata theoretic approach. To do this, we first introduce some notation. For a list of variables (Y_i, X_i), consider the quantification prefix $\check{\wp}_i \doteq \forall \check{X}_i \exists \check{Y}_i$ and then the quantification prefix $\wp_i \doteq \exists Y_1 \forall X_1 \ldots \exists Y_i \forall X_i \check{\wp}_i$. Intuitively, every quantification prefix $\wp_{i+1}$ is obtained from $\wp_i$ by pulling the existential quantification of Y_{i+1} up before the universal quantification of X_{i+1}. Clearly, we obtain that $\wp_0 = \forall X \exists Y$ and $\wp_n = \wp$. The automata construction builds on top of this quantifier transformation. First, recall that the satisfiability of $\wp_0 \psi$ amounts to solving the synthesis problem for ψ with X and Y being the variable blocks controlled by the environment and the agent, respectively. Let $\mathcal{A}_0$ be an alternating parity automaton that solves the synthesis problem, thus accepting 2^Y-labeled 2^X-trees representing the models of ψ. Now, for every $i < n$, define $\mathcal{A}_{i+1} \doteq \mathrm{change}_{2^{Y_i}, 2^{\hat{X}_i}}(\mathrm{ndet}(\mathcal{A}_i))$. We have the following.

Theorem 6 For every $i \leq n$, the formula $\wp_i \psi$ is satisfiable iff $\mathcal{L}(\mathcal{A}_i) \neq \emptyset$, where:

- $\mathcal{A}_0$ is the alternating parity automaton that solves the synthesis problem for ψ with agent variables Y and environment variables X, and
- $\mathcal{A}_{i+1} \doteq \mathrm{change}_{2^{Y_i}, 2^{\hat{X}_i}}(\mathrm{ndet}(\mathcal{A}_i))$, for every $i < n$.

Proof We prove by induction a stronger statement. We show that the automaton $\mathcal{A}_i$ accepts $2^{\check{Y}_i}$-labeled 2^X-trees θ_i for which there exist behavioral Skolem functions $\theta_1, \ldots, \theta_{i-1}$ such that $\theta_1 \oplus \ldots \oplus \theta_i$ is a behavioral Skolem function over $\wp_i$ that satisfies $\wp_i \psi$.

For the base case, observe that the behavioral Skolem functions that satisfy $\wp_0 \psi$ are all and only those strategies that solve the Synthesis for ψ. Now, since $\mathcal{A}_0$ accepts the 2^Y-labeled 2^X-trees that correspond to the winning strategies in the synthesis problem for ψ, we proved the statement.

For the induction case, assume that the statement is true for some i. Thus, the automaton $\mathcal{A}_i$ accepts $2^{\check{Y}_i}$-labeled 2^X-trees θ_i for which there exist $\theta_1, \ldots, \theta_{i-1}$ such that $\theta_1 \oplus \ldots \oplus \theta_i$ is a behavioral Skolem function that satisfies $\wp_i \psi$. From Theorem 5, it holds that the automaton $\mathcal{A}_{i+1} = \mathrm{change}_{2^{Y_i}, 2^{\hat{X}_i}}(\mathrm{ndet}(\mathcal{A}_i))$ accepts $2^{\check{Y}_{i+1}}$-labeled 2^X-trees θ_{i+1} that are in $\mathrm{shape}_{2^{Y_i+1}, 2^{\hat{X}_{i+1}}}(\mathcal{L}(\mathcal{A}_i))$.

Observe that, due to the application of $\mathrm{change}_{2^{Y_i}, 2^{\hat{X}_i}}$, the variable block Y_i depends only on $2^{\hat{X}_i}$-trees. Therefore, for every θ_{i+1} accepted in $\mathcal{A}_{i+1}$, we obtain that the composition $\theta_1 \oplus \theta_i \oplus \theta_{i+1}$ is a behavioral Skolem function over $\wp_{i+1}$ satisfying $\wp_{i+1} \psi$, proving the statement of the theorem. □

In Fig. 1 we depict the sequence of automata constructions employed in Theorem 6. Starting from formula $\wp_i\psi$, we build the automaton $\mathcal{A}_i$ that solves its satisfiability problem as follows. For $i = 0$, we observe that the satisfiability of $\wp_0\psi$ corresponds to the Synthesis of ψ, where the agent controls all variables of Y and the environment controls all the variables of X.

For $i > 0$, we walk through the schematics in the following order. First, we turn the formula into $\wp_{i-1}\psi$(take the left arrow from the current position), then we inductively build the automaton $\mathcal{A}_{i-1}$ that solves the Synthesis for $\wp_{i-1}$(take the down "snake" arrow below). Finally, by means of the operation $\text{change}_{2^{Y_i},2^{\hat{X}_i}} \circ \text{ndet}$(right arrow), we obtain the automaton desired automaton $\mathcal{A}_i$. The proof indeed essentially shows that such operation provides the desired construction. Note that, for each step right, transforming the automaton, we employ a nondetederminization, resulting in an exponential blow-up in the size of the automaton, increasing the complexity of Satisfiability for QLTL_B, as stated below.

Theorem 7 Satisfiability of a QLTL_B formula of the form $\varphi = \exists Y_1 \forall X_1 \ldots \exists Y_n \forall X_n \psi$ is $(n+1)$-EXPTIME-complete.

Proof From Theorem 6, we reduce the problem to the emptiness of the automaton $\mathcal{A}_n$, whose size is *n*-times exponential in the size of ψ, as we apply *n* times the nondeterminisation, starting from the automaton $\mathcal{A}_\psi$ that solves the synthesis problem for ψ. As the emptiness of the alternating parity automaton $\mathcal{A}_n$ involves another exponential blow-up, we obtain that the overall procedure is $(n+1)$-EXPTIME.

A matching lower-bound is obtained from the synthesis of distributed synthesis for hierarchically ordered architecture processes with LTL objectives, presented in [28], that is $(n+1)$-EXPTIME-complete with *n* being the number of processes. Indeed, every process p_i in such architecture synthesizes a strategy represented by a 2^{O_i}-labeled 2^{I_i}-tree, with O_i being the output variables and I_i the input variables. An architecture A is hierarchically ordered if $I_i \subseteq I_{i+1}$, for every process p_i. Thus, for an ordered architecture A and an LTL formula ψ, consider the variables $Y_i = O_i$ and $X_i = I_i \setminus I_{i-1}$ and the QLTL_B formula $\varphi = \exists Y_1 \forall X_1 \ldots \exists Y_n \forall X_n \psi$. A behavioral Skolem function θ that makes φ true corresponds to an implementation for the architecture that realizes (A, ψ). Moreover, the satisfiability of φ is $(n+1)$-EXPTIME, matching the lower-bound complexity of the realizability instance. □

We close this section by observing that the above techniques for solving QLTL_B satisfiability give us optimal techniques to solve conformant planning, contingent planing in FOND and contingent planing in PONDs in the case of LTL goals. Indeed for conformant planning we have to solve a formula of the form $\exists Y \forall X \varphi$ which belongs to $\Sigma_1^{\text{QLTL}_\text{B}}$ and can be solved in EXPSPACE. On the other hand conformant planning for LTL goals is EXPACE-complete

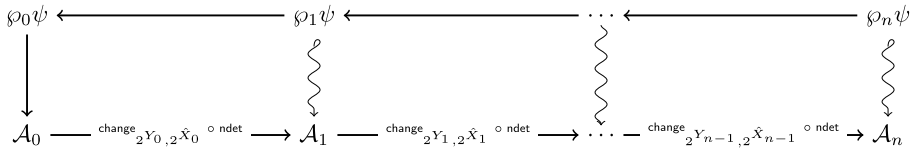

Fig. 1 Schematics of the automata construction sequence for Theorem 6

[42]. contingent planning in FOND is captured by a formula of the form $\forall X \exists Y \varphi$ which can be solved in 2-EXPTIME. On the other hand planning in FOND for LTL goals is 2-EXPTIME-complete–by reduction to synthesis [12]. Similarly, contingent planning in POND is captured by a formula of the form $\forall X_1 \exists Y \forall X_2 \varphi$, which although more complex than in the previous case still contains only a single block of the form $\forall X_i \exists Y_i$, and hence can still be solved in 2-EXPTIME. On the other hand planning in POND for LTL goals is 2-EXPTIME-complete–by reduction to synthesis under incomplete information [31].

Note also that this result gives us an optimal technique for solving synthesis and planing in nondeterministic domains for LTL goals. Indeed the QLTL_B formulas that capture them requires a single block of the form $\forall X_i \exists Y_i$, and hence satisfiability can be checked in 2-EXPTIME, thus matching the 2-EXPTIME-completeness of the two problems.

8 Weak-behavioral QLTL

We now introduce *weak-behavioral*QLTL, denoted QLTL_WB, that can be used to model systems in which *full observability* is forced over the execution histories. In such system every action performed by every player is *public*, meaning that it is visible to the entire system once it is occurred. In order to model this, we introduce an alternative definition of Skolem function, which we call here *weak-behavioral*.

We study satisfiability of QLTL_WB and show that its complexity is 2EXPTIME-complete via a reduction to a Multi-Player Parity Game [43] with a double exponential number of states and a (single) exponential number of colors.

Analogously to the case of QLTL_B, the logic QLTL_WB is defined in a Skolem-based approach.

Definition 5 For a given quantification prefix $\wp$ defined over a set $\text{Var}(\wp) \subseteq \text{Var}$ of propositional variables and a set $F = \text{Var} \setminus \text{Var}(\wp)$ of variables not occurring in $\wp$, a function $\theta : (2^{F \cup \forall(\wp)})^\omega \to (2^{\exists(\wp)})^\omega$ is a *weak-behavioral Skolem function* over $(\wp, F)$ if, for all $\pi_1, \pi_2 \in (2^{F \cup \forall(\wp)})^\omega$, $k \in \mathbb{N}$, and $Y \in \exists(\wp)$, it holds that

$$\theta(\pi_1)(0,k) = \theta(\pi_2)(0,k) \text{ and } \text{Prj}(\pi_1(k+1), \text{Dep}^F(Y)) = \text{Prj}(\pi_2(k+1), \text{Dep}^F(Y))$$
implies $\text{Prj}(\theta(\pi_1), Y) = \text{Prj}(\theta(\pi_2), Y)$.

In these Skolem functions, the evaluation of existential variables *Y* at every instant depends not only on the current evaluation of $\text{Dep}^F(Y)$ but also the evaluation history of each variable. The semantics of QLTL_WB is as follow.

Definition 6 A QLTL_WB formula $\varphi = \wp\psi$ is true over a trace π at an instant *i*, written $\pi, i \models_\text{WB} \varphi$, if there exists a weak-behavioral Skolem function θ over $(\wp, \text{free}(\varphi))$ such that $\hat{\theta}(\pi \Cup \pi'), i \models_\text{C} \psi$, for every $\pi' \in (2^{F \cup \forall(\wp)})^\omega$.

Differently from QLTL_B, QLTL_WB is not a special case of QLTL. As a matter of fact, they are incomparable. This is due to the fact that the existentially quantified variables depend, for standard Skolem functions, on the future of their dependencies, whereas, weak-behavioral functions, on the whole past of the computation, including the non-dependencies.

Consider again the formula $\varphi = \forall x \exists y(\mathrm{G}x \leftrightarrow y)$. This is not satisfiable as a $\mathrm{QLTL}_{\mathrm{WB}}$ formula, as this semantics still does not allow existential variables to depend on the future assignments of the universally quantified ones. On the other hand, the formula $\varphi = \exists y \forall x(\mathrm{F}x \leftrightarrow \mathrm{F}y)$ is satisfiable as a $\mathrm{QLTL}_{\mathrm{WB}}$. Indeed, the existentially quantified variable *y* can determine its value on an instant *i* by looking at the entire history of assignments, including those for *x*, although only on the past but not the present instant *i* itself. However, the semantics of both QLTL and $\mathrm{QLTL}_{\mathrm{B}}$ does not allow such dependence, which makes φ non satisfiable as both QLTL and $\mathrm{QLTL}_{\mathrm{B}}$.

Theorem 8 There exists a satisfiable $\mathrm{QLTL}_{\mathrm{B}}$ formula that is not satisfiable as $\mathrm{QLTL}_{\mathrm{WB}}$. Moreover, there exists a satisfiable $\mathrm{QLTL}_{\mathrm{WB}}$ formula that is not satisfiable as $\mathrm{QLTL}_{\mathrm{B}}$.

We now address satisfiability for $\mathrm{QLTL}_{\mathrm{WB}}$ by showing a reduction to multi-agent parity games [43]. Intuitively, a $\mathrm{QLTL}_{\mathrm{WB}}$ formula of the form $\varphi = \wp\psi$, with ψ being an LTL formula, establishes a multi-player parity game with ψ determining the parity acceptance condition and $\wp$ setting up the Player's controllability and team side. In order to present this result, we need some additional definitions.

An ω-word over an alphabet Σ is a special case of a Σ-labeled Υ-tree where the set of directions is a singleton. Being the set Υ irrelevant, an ω-word is also represented as an infinite sequence over Σ. The tree automata accepting ω-words are also called *word automata*. Word automata are a very useful way to (finitely) represent all the models of an LTL formula ψ. As a matter of fact, for every LTL formula ψ, there exists a deterministic parity word automaton $\mathcal{D}_\psi$ whose language is the set of traces on which ψ is true. The size of such automaton is double-exponential in the length of ψ. The following theorem gives precise bounds.

Theorem 9 ([44]). For every LTL formula ψ over a set Var of variables, there exists a *deterministic parity automaton* $\mathcal{D}_\psi = \langle 2^{\mathrm{Var}}, Q, q_0, \delta, \alpha \rangle$ of size double-exponential w.r.t. ψ and a (single) exponential number of priorities such that $\mathcal{L}(\mathcal{D}_\psi) = \{\pi \in (2^{\mathrm{Var}})^\omega \mid \pi \models_{\mathrm{C}} \psi\}$.

A multi-player parity game is a tuple $\mathcal{G} = \langle \mathrm{Pl}, (\mathrm{Ac}_i)_{i \in \mathrm{Pl}}, \mathrm{St}, s_0, \lambda, \mathrm{tr} \rangle$ where (i) $\mathrm{Pl} = \{0, \ldots, n\}$ is a set of players; (ii) Ac_i is a set of actions that player *i* can play; (iii) St is a set of states with s_0 being a designated initial state; (iv) $\lambda : \mathrm{St} \to C$ is a coloring function, assigning a natural number in *C* to each state of the game; (v) $\mathrm{tr} : \mathrm{St} \times \vec{\mathrm{Ac}} \to \mathrm{St}$, with $\vec{\mathrm{Ac}} = \mathrm{Ac}_1 \times \ldots \times \mathrm{Ac}_n$, is a transition function that prescribe how the game evolves in accordance with the actions taken by the players.

Players identified with an even index are the Even team, whereas the others are the Odd team. Objective of the Even team is to generate an infinite play over the set of states whose coloring fulfills the parity condition established by λ. A strategy for Player *i* of the Even team is a function $\mathrm{s}_i : \vec{\mathrm{Ac}}^* \times (\mathrm{Ac}_1 \times \mathrm{Ac}_{i-1}) \to \mathrm{Ac}_i$, that determines the action to perform in a given instant according to the past history and the current actions of players that perform their choices before *i*.

A tuple of strategies $\langle \mathrm{s}_0, \mathrm{s}_2, \ldots \rangle$ for the Even team is *winning* if every play that is generated by that, no matter what the Odd team responds, fulfills the parity condition.

Now, consider a QLTL$_{\mathrm{WB}}$ formula of the form $\varphi = \exists X_0 \forall X_1 \ldots \exists X_{n-1} \forall X_n \psi$, with X_0, X_n being possibly empty, and $\mathcal{D}_\psi = \langle 2^{\mathrm{Var}}, Q, q_0, \delta, \alpha \rangle$ being the DPW that recognizes the traces satisfying ψ, with $\alpha = \langle F_0, F_1, \ldots, F_k \rangle$. Then, consider the multi-player parity game $\mathcal{G}_\varphi = \langle \mathrm{Pl}, (\mathrm{Ac}_i)_{i \in \mathrm{Pl}}, \mathrm{St}, s_0, \lambda, \mathrm{tr} \rangle$ where (i) $\mathrm{Pl} = \{0, \ldots, n\}$; (ii) $\mathrm{Ac}_i = 2^{X_i}$ for each $i \in \mathrm{Pl}$; (iii) $\mathrm{St} = Q$ with $s_0 = q_0$; (iv) $\lambda : \mathrm{St} \to \mathbb{N}$ such that $\lambda(s) = \arg_j \{q \in F_j\}$; (v) $\mathrm{tr} = \delta$. The next theorem provides the correctness of this construction.

Theorem 10 A QLTL$_{\mathrm{WB}}$ formula φ is satisfiable iff there exists a winning strategy for the Even team in the multi-player parity game $\mathcal{G}_\varphi$.

Proof Observe that every player *i* is associated to the set of actions Ac_i corresponding to the evaluation of the variable block X_i. Also, every existential block of variables is associated to a player whose index is even and so playing for the Even team in $\mathcal{G}_\varphi$. In addition to this, the ordering of players reflects the order in the quantification prefix $\wp$.

Moreover, note that the strategy tuples for the Even team correspond to the weak-behavioral Skolem functions over $\wp$ and so they generate the same set of outcomes over $(2^X)^\omega$.

Since the automaton $\mathcal{D}_\psi$ accepts all and only those ω-words on which ψ is true, it follows straightforwardly that every weak-behavioral Skolem function θ over $\wp$ is such that $\theta(\pi) \models_{\mathrm{C}} \psi$ iff θ is a winning strategy for the Even team in $\mathcal{G}_\varphi$. Hence, the QLTL$_{\mathrm{WB}}$ formula φ is satisfiable iff $\mathcal{G}_\varphi$ admits a winning strategy for the Even team. □

Regarding the computational complexity of QLTL$_{\mathrm{WB}}$, consider that solving a multi-player parity game amounts to decide whether the Even team has a winning strategy in $\mathcal{G}$.

The complexity of solving a multi-player parity game $\mathcal{G}$ is polynomial in the number of states and exponential in the number of colors and players [43].

Therefore, we can conclude that the complexity of QLTL$_{\mathrm{WB}}$ satisfiability is as stated below.

Theorem 11 The complexity of QLTL$_{\mathrm{WB}}$ satisfiability is 2EXPTIME-complete.

Proof The procedure described in Theorem 10 is 2EXPTIME. Indeed, the automata construction of Theorem 9, produces a game whose set of states St is doubly-exponential in ψ and a number of colors *C* singly exponential in the size of ψ. Moreover, the number *n* of players in $\mathcal{G}_\varphi$ is bounded by the length of φ itself, as it corresponds to the number of quantifiers in the formula.

Now, from Theorem 4.1 in [43], we obtain that solving $\mathcal{G}_\varphi$ is polynomial in St, and exponential in both *C* and *n*. This amounts to a procedure that is double-exponential in the size of φ.

Regarding the lower-bound, observe that the formula $\forall X \exists Y \psi$ represents the synthesis problem for the LTL formula ψ with *X* and *Y* being the uncontrollable and controllable variables, which is already 2EXPTIME-complete [12]. □

We conclude the section by mentioning that, similarly to QLTL$_{\mathrm{WB}}$, an approach on restricting to behavioral quantifications while allowing full observability to all variables has been recently and independently taken in [45]. The authors first define a game semantics of QLTL in terms of *hyper-assignments* and a game over them with two players, *Verifier* and *Falsi-*

fier, in which the satisfiable formulas are those that allow for a winning strategy of Verifier. Then, they employ restrictions similar to the one of weak-behavioral Skolem functions, thus providing the quantifications of the logic with a strategic flavor.

It is important to notice that their approach, as QLTL_{WB}, are significantly different from QLTL_{B}.

In particular, the full visibility on the history of evaluations does not admit a direct application in planning and synthesis, as discussed in Section 4. It is such full visibility that simplifies the logic and allows for lower complexity in these variants, as it is shown by Theorem 11.

9 Relationship with behavioral strategy logic

The notion of behavioral has recently drawn the attention of many researchers in the area of logic for strategic reasoning. Strategy Logic [21] (SL) has been introduced as a formalism for expressing complex strategic and game-theoretic properties. Strategies in SL are first class citizens. Unfortunately, and similarly to QLTL, quantifications over them sets up a kind of dependence that cannot be realized through actual processes, as they involve future and counter-factual possible computations that are not accessible to reactive programs. To overcome this, and also mitigate the computational complexities of the main decision problems, the authors introduced a *behavioral* semantics as a way to restrict the dependence among strategies to a realistic one. They also showed that for a small although significant fragment of SL, which includes ATL*, behavioral semantics has the same expressive power of the standard one. This means that "*behavioral strategies*" are able to solve the same set of problems that can be expressed in such fragment. Further investigations around this notion has been carried out in the community. In [23, 24], the authors characterize different notions of behavioral, ruling out future and counter-factual dependence one by one, providing a classification of syntactic fragments for which the behavioral and non-behavioral semantics are equivalent.

In this section, we compare QLTL_{B} and QLTL_{WB} with behavioral Strategy Logic [21].

Formally, the syntax of SL is defined as an extension of LTL with two strategy quantifiers, existential ($\langle\!\langle x \rangle\!\rangle$) and universal ($[\![x]\!]$) and a binding predicate (a, x) assigning a strategy x to an agent a. Below, the formal definition of SL syntax is reported:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi \mid \langle\!\langle x \rangle\!\rangle \varphi \mid [\![x]\!]\varphi \mid (a, x)\varphi$$

where $p \in \mathrm{AP}$ belongs to a (finite) set of *atomic propositions* $x \in \mathrm{Var}$ is a first-order variable, and $a \in \mathrm{Agn}$ is an agent.

Formulas of SL are interpreted over *Concurrent Game Structures* (CGS) [22], that can be seen as extensions of Kripke structures [46] in which transitions are (collectively and concurrently) controlled by a (finite) set Agn of agents, each of them performing an action at each step of the computation. Formally, a CGS is a tuple of the form $\mathcal{G} = \langle \mathrm{St}, \mathrm{Agn}, \mathrm{Ac}, s_0, \lambda, \mathrm{tr} \rangle$, where (i) St is a finite set of (global) *states*; (ii) Agn is a finite set of *agents*; (iii) Ac is a finite set of *actions*; (iv) s_0 is a designated *initial state*; (v) $\lambda : \mathrm{St} \to 2^{\mathrm{AP}}$ labels each state with an evaluation of the atomic propositions; (vi) $\mathrm{tr} : \mathrm{St} \times \mathrm{Ac}^{\mathrm{Agn}} \to \mathrm{St}$ is a (deterministic) transition function, assigning successor states according to the aggregated action of the agents.

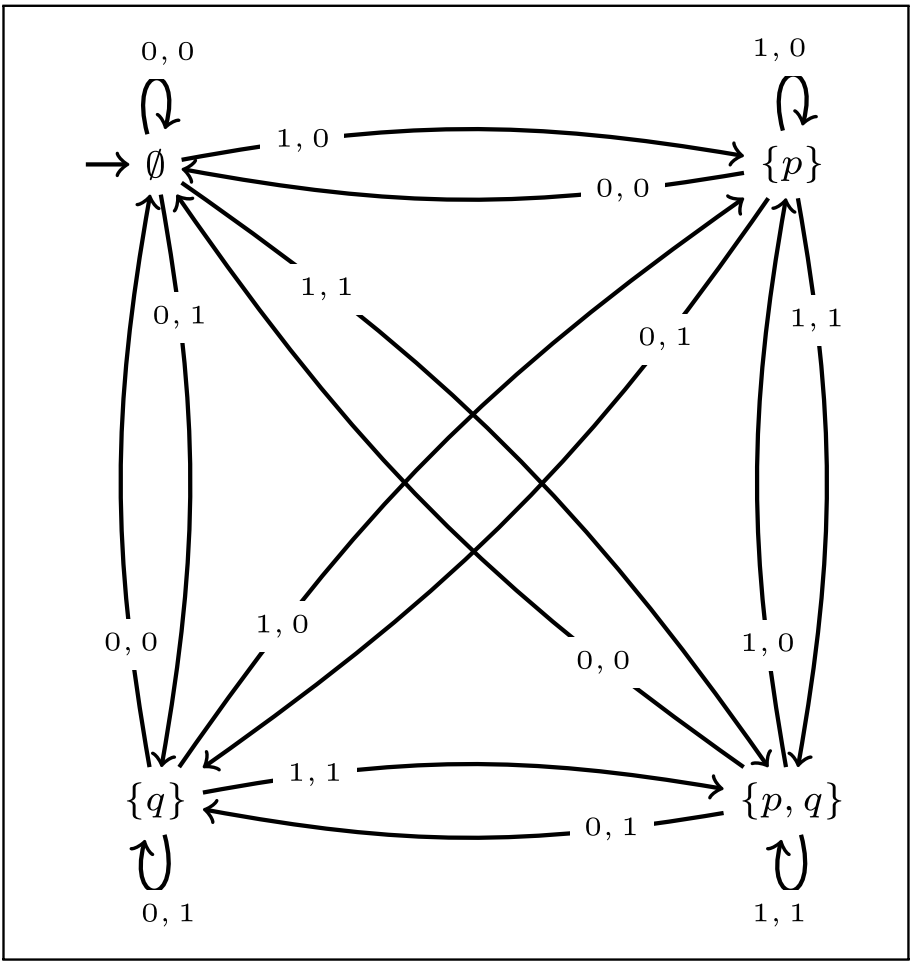

Fig. 2 A representation of a CGS with two agents, each of one controlling the truth value of the atomic propositions p and q, respectively

Starting from s_0, the system evolves to a next state according to the actions taken (individually and independently) by the agents, following the transition function tr. The infinite sequence resulting from this mechanism, once evaluated over the labeling function λ, determines a trace over AP, over which the temporal part of a SL formula is evaluated.

In order to perform their actions, agents make use of *strategies* that are defined as functions of the form $\mathrm{s} : \mathrm{St}^* \to \mathrm{Ac}$. Strategies so defined are central to the definition of the semantics in SL, whose details can be found in [21].

Importantly, strategies in the semantics of SL are **kept visible** to every agent in play, at any time. This assumption comes from the fact that an action in s is chosen over sequences of (global) states carrying out information about the behavior of every agent in the system. Consider for example the CGS $\mathcal{G}$ represented in Fig. 2, with two agents, namely a_p and a_q, each of them using actions 0, 1 to set the truth-value of a singly-controlled variable p and q, respectively. Consider the SL formula $\varphi = \langle\langle y \rangle\rangle [\![x]\!](a_p, y)(a_q, x)(\mathrm{F}p \leftrightarrow \mathrm{F}q)$. It holds that

$\mathcal{G} \models \varphi$. As a matter of fact, a strategy s_y for agent a_p that takes 1 on every history that visited either $\{q\}$ or $\{p, q\}$ at least once is enough to satisfy the temporal part, no matter which strategy s_x is taken by agent a_q.

Notice that φ is also in SL[1G], a fragment of SL that considers only formulas where every quantification block is followed by a binding block mentioning every agent in the game structure. In such fragment the classic and behavioral semantics coincide [21, Thm 4.2]. Such invariance result in SL[1G] is also central to prove that the complexity of both its model-checking and satisfiability is significantly easier than the entire logic. As a matter of fact, both of them are proved to be 2EXPTIME-complete [21, 47], in contrast with the non-elementary and undecidability of the corresponding decision problem for full SL.

Compare now with the formula $\exists y \forall x (\mathrm{F} x \leftrightarrow \mathrm{F} y)$. This is not satisfiable neither as a QLTL nor as a $\mathrm{QLTL_B}$ formula. The reason why is that the process controlling the truth-value of y does not have the visibility over x at any point in time during the computation, and so it is not capable of reacting appropriately to its truth-value evolution. In other words, the process for y will never see whether x has become true at some instant, and so cannot correctly set up the value of y for the entire (infinite) execution, accordingly. Note that this is not only an issue of non-behavioral semantics, but also related to the visibility of variables along the execution. On the other hand, and similarly to SL[1G], if we read the formula as $\mathrm{QLTL_{WB}}$, this becomes satisfiable, as the process controlling y will always have visibility on the past values of x, and so will be able to cast y true on the moment it sees x being true at least once in the history of the computation.

With this example, we argue that the simplification in complexity of the decision problems for SL[1G] is not only due to the behavioral property, but also intrinsic in the fact that the semantics provides full visibility to the agents in play. Notice that our approach for $\mathrm{QLTL_{WB}}$ satisfiability could be also used, *mutatis mutandis*, to provide an alternative (and computational complexity optimal) technique for the model checking of SL[1G]. Indeed, it is not hard to see how the problem can be approached by turning the temporal part into a parity automaton and then cast the corresponding concurrent multi-parity game obtained from the intersection with the CGS involved.

10 Conclusion

We introduced a behavioral semantics for QLTL, getting a new logic *Behavioral* QLTL ($\mathrm{QLTL_B}$). This logic is characterized by the fact that the (second-order) existential quantification of variables is restricted to depend, at every instant, only on the past truth assignments of the variables that are universally quantified upfront in the formula, and not on their entire trace, as it is for classic QLTL. This makes such dependence to be a function readily implementable by processes, thus making $\mathrm{QLTL_B}$ suitable for capturing advanced forms of planning and synthesis through standard reasoning, as envisioned since in the early days of AI [2]. We studied satisfiability for $\mathrm{QLTL_B}$, providing tight complexity bounds. For the simplest syntactic fragments, which do not include quantification blocks of the form $\forall X_i \exists Y_i$, the complexity is the same as QLTL, given that the two semantics are equivalent. For the rest of $\mathrm{QLTL_B}$, where the characteristics of behavioral semantics become apparent, we present an automata-based technique that is $(n+1)$-EXPTIME, with n being the number of quantifi-

cation blocks $\forall X_i \exists Y_i$. A matching lower-bound comes from a reduction of the corresponding (distributed) synthesis problems.

We also considered a weaker-version of Behavioral QLTL, denoted QLTL_{WB}, where the history of quantification is completely visible to every existentially quantified variable, except for the current instant in which only the upfront quantification is available. We showed that satisfiability is 2-EXPTIME, regardless of the number of quantifications in the formula. This is due to the fact that full visibility of variables allows for solving the problem with a simple local reasoning that avoids computationally expensive automata constructions. A matching lower-bound comes again from a reduction of a corresponding synthesis problem.

Recently, Linear Temporal Logic on finite traces (LTL $_f$) has been considered [48] and its decision problems investigated, including different kinds of synthesis [49, 50], as well as their counterparts in planning [13, 51]. Despite their decision problems have the same complexity as its infinite trace counterparts, LTL $_f$ has proven to be much more well-behaved from the computational point of view [52–54]. The crux of the advantage of using LTL_f is that the determinization procedure needed for synthesis is the standard subset construction [55] which has good computational properties in practice [56, 57]. This contrasts with the determinization of Büchi automata for which instead no scalable algorithm exists yet [58]. Now, it is of interest to introduce and investigate BQLTL $_f$, the finite trace version of QLTL, still with a behavioral semantics. The solution techniques presented here could in principle be used also for the finite trace setting and the algorithm used to get the complexity upper-bound in this paper could become a practical algorithm for actually reasoning with BQLTL $_f$. We plan to study this in the future.

Acknowledgments This work is supported in part by the ERC Advanced Grant WhiteMech (No. 834228), the PRIN project RIPER (No. 20203FFYLK) and the PRIN project PINPOINT (No. B87G22000450001), the PNRR MUR project FAIR (No.PE0000013), the UKRI Erlangen AI Hub on Mathematical and Computational Foundations of AI, and the Sapienza University of Rome under the Progetti Grandi di Ateneo programme, grant RG123188B3F7414A (ASGARD - Autonomous and Self-Governing Agent-Based Rule Design).

Author Contributions All authors contributed equally to this manuscript.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

References

1. Abadi, M., Lamport, L., & Wolper, P. (1989). Realizable and unrealizable specifications of reactive systems. In: *ICALP'89. vol. 372 of LNCS* (pp. 1–17). Springer.
2. Green, C.C. (1969). Application of theorem proving to problem solving. In: *IJCAI'69* (pp. 219–240).
3. Barringer, H., Fisher, M., Gabbay, D. M., Gough, G., & Owens, R. (1995). METATEM: An introduction. *Formal Aspects of Computing,7*(5), 533–549.
4. Bacchus, F., & Kabanza, F. (1998). Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence,22*(1–2), 5–27.
5. Bacchus, F., & Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. *Artificial Intelligence,116*(1–2), 123–191.

6. Cimatti, A., Giunchiglia, F., Giunchiglia, E., Traverso, P. (1997). Planning via model checking: A decision procedure for *AR*. In: *ECP'97 vol. 1348 of LNCS*. p. 130–142.
7. Cimatti, A., & Roveri, M. (2000). Conformant planning via symbolic model checking. *J Artif Intell Res.,13*, 305–338.
8. Daniele, M., Traverso, P., & Vardi, M.Y. (1999). Strong cyclic planning revisited. In: *ECP'99. vol. 1809 of LNCS* (pp. 35–48). Springer.
9. Bertoli, P., Cimatti, A., & Roveri, M. (2001). Heuristic search + symbolic model checking = efficient conformant planning. In: *IJCAI'01* (pp. 467–472).
10. Cerrito, S., & Mayer, M. (1998) Bounded model search in linear temporal logic and its application to planning. In: *TABLEAUX'98. vol. 1397 of LNCS* (pp. 124–140). Springer.
11. Calvanese, D., De Giacomo, G., & Vardi, M.Y. (2002). Reasoning about Actions and Planning in LTL Action Theories. In: *KR'02* (pp. 593–602).
12. Pnueli, A., & Rosner, R. (1989). On the synthesis of a reactive module. In: *POPL*(pp. 179–190). ACM.
13. Camacho, A., Bienvenu, M., & McIlraith, S. (2019). Towards a unified view of AI planning and reactive synthesis. In: *ICAPS* (pp. 58–67). AAAI Press.
14. Aminof, B., De Giacomo, G., Murano, A., & Rubin, S. (2019). Planning under LTL environment specifications. In: *ICAPS* (pp. 31–39). AAAI Press.
15. Chatterjee, K., & Henzinger, T.A. (2007). Assume-guarantee synthesis. In: *TACAS'07. vol. 4424 of LNCS* (pp. 261–275).
16. Chatterjee, K., Henzinger, T.A., & Jobstmann, B. (2008). Environment Assumptions for Synthesis. In: *CONCUR'08* (pp. 147–161).
17. Church, A. (1963). Logic, arithmetics, and automata. In: *Proc. Int. Congress of Mathematicians, 1962* (pp. 23–35).
18. Sistla, A. P., Vardi, M. Y., & Wolper, P. (1987). The complementation problem for Büchi automata with applications to temporal logic. *TCS.,49*, 217–237.
19. Rintanen, J. (2004). Complexity of planning with partial observability. In: *ICAPS'04* (pp. 345–354).
20. Chatterjee, K., Henzinger, T. A., & Piterman, N. (2010). Strategy logic. *Information and Computation,208*(6), 677–693. https://doi.org/10.1016/j.ic.2009.07.004
21. Mogavero, F., Murano, A., Perelli, G., & Vardi, M. (2014). Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic,15*(4), 34:1-34:47.
22. Alur, R., Henzinger, T. A., & Kupferman, O. (2002). Alternating-time temporal logic. *The Journal of the ACM,49*(5), 672–713.
23. Gardy, P., Bouyer, P., & Markey, N. (2018). Dependences in Strategy Logic. In: *STACS'18 vol. 96 of LIPIcs* (pp. 34:1–34:15).
24. Gardy, P., Bouyer, P., & Markey, N. (2020). Dependences in strategy logic. *Theory of Computing Systems,64*(3), 467–507.
25. Fijalkow, N., Maubert, B., Murano, A., Rubin, S., & Vardi, M.Y. (2022). Public and private affairs in strategic reasoning. In: Kern-Isberner, G., Lakemeyer, G., & Meyer, T., (Eds.), *KR'22*. Available from: https://proceedings.kr.org/2022/14/.
26. Cimatti, A., Roveri, M., & Traverso, P. (1998). Strong planning in non-deterministic domains via model checking. In: *AIPS* (pp. 36–43). AAAI
27. Geffner, H., & Bonet, B. (2013). A concise introduction to models and methods for automated planning. Morgan & Claypool.
28. Pnueli, A., & Rosner, R. (1990). Distributed reactive systems are hard to synthesize. In: *FOCS'90* (pp. 746–757).
29. Kupferman, O., & Vardi, M.Y. (2001). Synthesizing distributed systems. In: *LICS'01* (pp. 389–398).
30. Finkbeiner, B., & Schewe, S. (2005). Uniform Distributed Synthesis. In: *LICS'05* (pp. 321–330).
31. Kupferman, O., & Vardi, M.Y. (2000). In: Synthesis with incomplete informatio (pp. 109–127). Springer.
32. Mogavero, F., Murano, A., & Vardi, M. (2010). Reasoning about strategies. In: Lodaya, K., Mahajan, M. (Eds.), *FSTTCS. vol. 8 of LIPIcs* (pp. 133–144). Schloss Dagstuhl - Leibniz-Zentrum für Informatik. Available from: https://doi.org/10.4230/LIPIcs.FSTTCS.2010.133.
33. Sistla, A.P. (1985). Theoretical issues in the design and verification of distributed systems. PhD Thesis.
34. De Giacomo, G., & Perelli, G. (2023). Behavioral QLTL. In: Malvone, V., Murano, A., editors *Multi-agent systems - 20th European Conference, EUMAS 2023, Naples, Italy, September 14-15, 2023, Proceedings. vol. 14282 of Lecture Notes in Computer Science* (pp. 133–149). Springer. Available from: https://doi.org/10.1007/978-3-031-43264-4_9
35. Pnueli, A. (1977). The Temporal Logic of Programs. In: *FOCS-77* (pp. 46–57).
36. Gabbay, D.M., Pnueli, A., Shelah, S., & Stavi, J. (1980). On the temporal basis of fairness. In: Abrahams, P.W., Lipton, R.J., Bourne, S.R. & (Eds.), *POPL'80* (pp. 163–173).

37. Thomas, W. (1997). Languages, automata, and logic. In: *Handbook of formal languages, volume 3: beyond words* (pp. 389–455). Springer
38. Ghallab, M., Nau, D.S., & Traverso, P. (2004). Automated planning - theory and practice. 1st ed. (pp. 635). Elsevier
39. Goldman, R.P., & Boddy, M.S. (1996). expressive planning and explicit knowledge. In: *Proc. of AIPS*.
40. Aminof, B., De Giacomo, G., & Rubin, S. (2020). Stochastic fairness and language-theoretic fairness in planning in nondeterministic domains. In: *ICAPS* (pp. 20–28). AAAI Press.
41. Muller, D.E., & Schupp, P.E. (1984). Alternating automata on infinite objects, determinacy and Rabin's theorem. In: *Automata on infinite words. vol. 192 of LNCS* (pp. 100–107). Springer
42. De Giacomo, G., & Vardi, M. (1999). Automata-theoretic approach to planning for temporally extended goals. In: *ECP. vol. 1809 of Lecture notes in computer science* (pp. 226–238). Springer
43. Malvone, V., Murano, A., Sorrentino, L. (2016). Concurrent multi-player parity games. In: *AAMAS'16* (pp. 689–697).
44. Piterman, N. (2007). From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science,3*(3).
45. Bellier, D., Benerecetti, M., Della Monica, D., Mogavero, F. (2021). Good-for-Game QPTL: An alternating hodges semantics. arXiv:2104.06085
46. Kripke, S. A. (1963). Semantical considerations on modal logic. *APF,16*, 83–94.
47. Mogavero, F., Murano, A., Perelli, G., & Vardi, M. (2017). Reasoning about strategies: On the satisfiability problem. *Logical Methods in Computer Science.,13*(1). https://doi.org/10.23638/LMCS-13(1:9)2017
48. De Giacomo, G., & Vardi, M.Y. (2013). Linear temporal logic and linear dynamic logic on finite traces. In: *IJCAI* (pp. 854–860).
49. De Giacomo, G., & Vardi, M.Y. (2015). Synthesis for LTL and LDL on finite traces. In: *IJCAI* (pp. 1558–1564).
50. De Giacomo, G., & Vardi, M.Y. (2016). LTL[CDATA[_{\text{f}}]] and LDL $_{\text{f}}$ [CDATA[_{\text{ f }}]] Synthesis under partial observability. In: *IJCAI* (pp. 1044–1050).
51. Camacho, A., Triantafillou, E., Muise, C., Baier, J., & McIlraith, S. (2017). Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In: Singh, S.P., & Markovitch, S. (Eds.) (pp. 3716–3724). AAAI
52. Zhu, S., Tabajara, L., Li, J., Pu, G., & Vardi, M. (2017). Symbolic LTL$_f$[CDATA[_f]] synthesis. In: *IJCAI* (pp. 1362–1369).
53. Bansal, S., Li, Y., Tabajara, L., & Vardi, M. (2020). Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications. In: *AAAI* (pp. 9766–9774).
54. De Giacomo, G., & Favorito, M. (2021). Compositional approach to translate LTLf/LDLf into deterministic finite automata. In: *ICAPS* (pp. 122–130). AAAI Press
55. Rabin, M., & Scott, D. (1959). Finite automata and their decision problems. *IBM Journal of Research and Development,3*(2), 114–125. https://doi.org/10.1147/rd.32.0114
56. Tabakov, D., Vardi, M. (2005). Experimental evaluation of classical automata constructions. In: *LPAR vol. 3835 of Lecture Notes in Computer Science* (pp. 396–411). Springer
57. Tabakov, D., Rozier, K., & Vardi, M. (2012). Optimized temporal monitors for SystemC. *Formal Methods in System Design,41*(3), 236–268.
58. Finkbeiner, B. (2016). Synthesis of reactive systems. *Dependable Software Systems Engineering,45*, 72–98.