Behavioral QLTL

Giuseppe De Giacomo[1,2] and Giuseppe Perelli[2]

[1] University of Oxford
[2] Sapienza University of Rome

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" [1]. 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. While as this 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.

1 Introduction

Since the very early time of AI, researchers have tried to reduce planning to logical reasoning, i.e., satisfiability, validity, logical implication [29]. 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 Linear 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 [7], the work on temporally extended goals and declarative control constraints [5, 6], the work on planning via model-checking [17, 18, 20, 9], the work on adopting LTL logical reasoning (plus some meta-theoretic manipulation) for certain forms of planning [12, 10]. More recently the connection between planning in nondeterministic domains and (reactive) synthesis [35] 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 [11, 3], also related to synthesis under assumptions [13, 14].

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*" [16,

35], 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) [40] and then use its classic reasoning tasks, such as satisfiability, validity and logical implication, to capture planning and synthesis?

In [10] a positive answer was given limited to conformant planning [37], 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. [10] 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) [37]. 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) [15, 32]. This has led to investigating the "*behavioral*" semantics in these logics, i.e., a semantics based on processes. In their seminal work [32], 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 ATL* [2], 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 [24, 25, 33].

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$_\mathsf{B}$), 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$_\mathsf{B}$

perfectly suitable to capture extended forms of planning and synthesis through standard reasoning tasks (satisfiability in particular).

In QLTL$_\mathsf{B}$, 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$_\mathsf{B}$ 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 [37]. 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$_\mathsf{B}$ formula $\forall X \exists Y \psi$ represents contingent planning in fully observable domains [37], also known as *Strong Planning in Fully Observable Nondeterministic Domains* (FOND) [19, 26], as well as LTL synthesis [35]. The QLTL$_\mathsf{B}$ formula $\forall X_1 \exists Y \forall X_2 \varphi$ represents the problem of contingent planning under partial observability [37], also known as *Strong Planning in Partially Observable Nondeterministic Domains* (POND) [26]. 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 [36, 31, 22].

We study QLTL$_\mathsf{B}$ 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$_\mathsf{B}$ 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 [35, 30].

2 Quantified Linear Temporal Logic

Linear Temporal Logic (LTL) was originally proposed in Computer Science as a specification language for concurrent programs [34]. Formulas of LTL are built from a set $\mathtt{Var}$ of *propositional variables* (or simply variables), together with Boolean and temporal operators. Its syntax can be described as follows:

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

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

Intuitively, the formula $\mathtt{X}\psi$ says that ψ holds at the *next* instant. Moreover, the formula $\psi_1 \mathtt{U} \psi_2$ says that at some future instant ψ_2 holds and *until* that point, ψ_1 holds. We also use the standard Boolean abbreviations $\mathtt{true} := x \vee \neg x$ (*true*), $\mathtt{false} := \neg\mathtt{true}$ (*false*), and $\psi_1 \rightarrow \psi_2 := \neg\psi_1 \vee \psi_2$ (*implication*). In addition, we also use the binary operator $\psi_1 \mathtt{R} \psi_2 \doteq \neg(\neg\psi_1 \mathtt{U} \neg\psi_2)$ (*release*) and the unary operators $\mathtt{F}\psi := \mathtt{true}\mathtt{U}\psi$ (*eventually*) and $\mathtt{G}\psi := \neg\mathtt{F}\neg\psi$ (*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* $\pi \in (2^{\mathtt{Var}})^\omega$ is an infinite sequence of truth assignments over the set of variables $\mathtt{Var}$, where $(\cdot)^\omega$ is the classic omega operator used to denote such infinite sequences. By $\pi(i) \in 2^{\mathtt{Var}}$, 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 $\pi(i, j) \doteq \pi(i), \ldots, \pi(j) \in (2^{\mathtt{Var}})^*$ 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 $X \subseteq \mathtt{Var}$ 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 $\mathtt{Var} \setminus X$, whenever the set $\mathtt{Var}$ is clear from the context.
We say that an LTL formula ψ is true on an assignment π at instant i, written $\pi, i \models_{\mathtt{LTL}} \psi$, if:

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

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

Quantified Linear-Temporal Logic (QLTL) is an extension of LTL with two *Second-order* quantifiers [39]. 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:

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

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

Note that this is a proper extension of LTL, as QLTL has the same expressive power of MSO [39], whereas LTL is equivalent to FOL [23].

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

Observe that we can reverse the projection operation by combining traces over disjoint sets of variables. More formally, for two disjoint sets $X, X' \subseteq \mathtt{Var}$ 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 $\pi, i \models_{\mathtt{C}} \varphi$, is defined as follows:

- $\pi, i \models_{\mathtt{C}} \psi$ iff $\pi, i \models_{\mathtt{LTL}} \psi$ for every quantifier-free (LTL) formula ψ;
- $\pi, i \models_{\mathtt{C}} \exists x\varphi$ iff there exists π' agreeing with π over $-x$ s.t. $\pi', i \models_{\mathtt{C}} \varphi$;
- $\pi, i \models_{\mathtt{C}} \forall x\varphi$ iff for each π' agreeing with π over $-x$, it holds that $\pi', i \models_{\mathtt{C}} \varphi$;

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 $\mathsf{free}(\varphi)$ we denote the set of free variables in φ.

As for LTL, we say that φ is true on π, and write $\pi \models_{\mathtt{C}} \varphi$ iff $\pi, 0 \models_{\mathtt{C}} \varphi$. 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 $\mathsf{free}(\varphi)$ 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 $\mathtt{Var}$. Hence a QLTL formula in *prenex normal form* has the form $\wp\psi$, where $\wp = \mathtt{Qn}_1 x_1 \dots \mathtt{Qn}_n x_n$ is a *prefix quantification* with $\mathtt{Qn}_i \in \{\exists, \forall\}$ 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 $\mathtt{G}(\exists y(y \wedge \mathtt{X}\neg y))$. This is equivalent to $\forall x \exists y(\mathtt{singleton}(x) \rightarrow (\mathtt{G}(x \rightarrow (y \wedge \mathtt{X}\neg y))))$, with $\mathtt{singleton}(x) \doteq \mathtt{F}x \wedge \mathtt{G}(x \rightarrow \mathtt{XG}\neg x)$ expressing the fact that x is true exactly once on the trace [3]. A full proof of the reduction to prenex normal form can be found in [41, 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 existential 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 [39], 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 $\mathtt{Qn}_1 X_1 \ldots \mathtt{Qn}_n X_n \psi$ 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 $\mathtt{Qn}_i = \exists$ iff $\mathtt{Qn}_{i+1} = \forall$, 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

- $\pi, i \models_{\mathtt{C}} \psi$ iff $\pi, i \models_{\mathtt{LTL}} \psi$, for every quantifier-free formula ψ;
- $\pi, i \models_{\mathtt{C}} \exists X\varphi$ iff there exists π' agreeing with π over $-X$ s.t. $\pi', i \models_{\mathtt{C}} \varphi$;
- $\pi, i \models_{\mathtt{C}} \forall X\varphi$ iff for each π' agreeing with π over $-X$, it holds that $\pi', i \models_{\mathtt{C}} \varphi$ [4].

From now on, we might refer to variable blocks, simply as blocks. Moreover, with a slight overlap of notation, we write $X \subseteq \exists(\wp)$ 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

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

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

only if, $\exists \mathsf{free}(\varphi)\varphi$ 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 $\varphi = \exists y(y \leftrightarrow \mathtt{G}x)$ with $\mathsf{free}(\varphi) = \{x\}$. 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 $(y \leftrightarrow \mathtt{G}x)$. Notice that $\varphi = \exists y(y \leftrightarrow \mathtt{G}x)$ is satisfiable if and only if the close formula $\exists x \exists y(y \leftrightarrow \mathtt{G}x)$ is so. Analogously, φ is valid if and only if the close formula $\forall x \exists y(y \leftrightarrow \mathtt{G}x)$ is so.

Such problem is decidable, though computationally highly intractable in general [39]. For a given natural number k, by k-EXPSPACE we denote the language of problems solved 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 ([40]). *Satisfiability for k-QLTL 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 $\mathsf{Dep}_\wp(Y) = \{X \in \forall(\wp) \mid X <_\wp Y\}$ we denote the blocks to which Y depends on in $\wp$. Moreover, for a given set $F \subseteq \mathtt{Var}$ of variables, sometimes referred as the *free variables block*, by $\mathsf{Dep}^F_\wp(Y) = F \cup \mathsf{Dep}_\wp(Y)$ we denote the *augmented dependency*, taking into account the additional free block. Whenever clear from the context, we omit the subscript and simply write $\mathsf{Dep}(Y)$ and $\mathsf{Dep}^F(Y)$.

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 $\mathtt{Var}(\wp) \subseteq \mathtt{Var}$ of variables, and a free block $F = \mathtt{Var} \setminus \mathtt{Var}(\wp)$, 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 $\mathtt{F} \cup \forall(\wp)$ and for all blocks $Y \in \exists(\wp)$, it holds that

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

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 $\pi, i \models_{\mathtt{S}} \varphi$, if there exists a Skolem function θ over $(\wp, \mathsf{free}(\varphi))$ such that $\hat{\theta}(\pi \uplus \pi_{\forall(\wp)}), i \models_{\mathtt{LTL}} \psi$, 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 $\pi \models_{\mathtt{S}} \varphi$ if, and only if, $\pi \not\models_{\mathtt{S}} \neg\varphi$.

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

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

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 $\pi_F \models_{\mathtt{C}} \psi$ iff $\pi_F \models_{\mathtt{S}} \psi$.

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

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

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

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

4 Behavioral QLTL ($\mathbf{QLTL_B}$)

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 $\forall x \exists y(\mathtt{G}x \leftrightarrow y)$. 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., [32, 25].

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 $\text{QLTL}_{\mathtt{B}}$, 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 $\mathtt{Var}$. However, while the syntax is the same of QLTL, the semantics of $\text{QLTL}_\mathtt{B}$ is defined in terms of behavioral Skolem functions.

Definition 3 (Behavioral Skolem function). *For a given quantification prefix $\wp$ defined over a blocks of propositional variables $\mathtt{Var}$ 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*

$$\mathsf{Prj}(\pi_1(0,k), \mathsf{Dep}^F(Y)) = \mathsf{Prj}(\pi_2(0,k), \mathsf{Dep}^F(Y))$$
$$\textit{implies}$$
$$\mathsf{Prj}(\theta(\pi_1)(0,k), Y) = \mathsf{Prj}(\theta(\pi_2)(0,k), Y).$$

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 $\text{QLTL}_\mathtt{B}$ as follows.

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

A $\text{QLTL}_\mathtt{B}$ formula φ is true on a trace π, written $\pi \models_\mathtt{B} \varphi$, if $\pi, 0 \models_\mathtt{B} \varphi$. 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 $\varphi = \exists y(y \leftrightarrow \mathtt{G}x)$ with $\mathsf{free}(\varphi) = \{x\}$, now in $\text{QLTL}_\mathtt{B}$. This is satisfiable again. Indeed, consider the behavioral Skolem function θ such that $\theta(\pi_x)(0,0) = \mathtt{true}$ and $\theta(\pi_x)(0,k) = \mathtt{false}$ 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 $(y \leftrightarrow \mathtt{G}x)$.

Again, notice that $\varphi = \exists y(y \leftrightarrow \mathtt{G}x)$ is satisfiable if and only if the close formula $\exists x \exists y(y \leftrightarrow \mathtt{G}x)$ is so. Indeed, now notice that the Skolem function chose both the values of x and y as needed in $(y \leftrightarrow \mathtt{G}x)$. However, the formula φ is not valid. Indeed, the closed formula $\forall x \exists y(y \leftrightarrow \mathtt{G}x)$ is neither satisfiable nor valid in $\text{QLTL}_\mathtt{B}$ 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 $\forall x \exists y(y \leftrightarrow \mathtt{G}x)$ is not satisfiable. Indeed, the formula $\exists y \forall x(x \not\leftrightarrow \mathtt{G}y)$ 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 [32, 25].

Now, consider the formula $\varphi = \exists y \mathtt{G}(y \leftrightarrow x)$. This is both satisfiable and valid. Indeed, in the case of satisfiability, the closed formula $\exists x \exists y \mathtt{G}(y \leftrightarrow x)$ is satisfiable as the behavioral Skolem function can chose the values of x and y appropriately. For the case of validity, the closed formula $\forall x \exists y \mathtt{G}(y \leftrightarrow x)$ 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 $\exists y \forall x \mathtt{G}(y \leftrightarrow x)$ 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$_\mathsf{B}$, it is interesting to see how QLTL$_\mathsf{B}$ can capture advanced forms of Planning. We assume some familiarity with Planning in AI, see [27, 26]. 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 [5]. 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 [10, 3]. 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$_\mathsf{B}$. 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$_\mathsf{B}$ 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* [18]. 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)* [19, 26]. 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) [35].

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 [28, 26]. 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) [30]. Notice also that we can indeed include fairness assumptions in φ_D and hence in φ, so with some care, see [4], the above two QLTL$_B$ formulas can capture also strong cyclic plans [20, 26].

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 POND planning.

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* [36, 22]. 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 [36]. However, it has been proven that the absence of information forks is sufficient to guarantee the decidability of synthesis [22]. Specifically, without information forks it is possible to arrange the agents in a sort of information hierarchy, which leads to decidability [22]. Incidentally, this is the form of uniform distributed synthesis that is captured by the above QLTL$_\mathsf{B}$ formulas. Indeed we will show later that solving a distributed synthesis with hierarchical information architectures can be done optimally by reduction to QLTL$_\mathsf{B}$ satisfiability of the formulas presented above.

6 QLTL$_B$ Properties

Clearly, since QLTL$_\mathsf{B}$ 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$_\mathsf{B}$ formula φ is equivalent to the one of $\exists\mathsf{free}(\varphi)\varphi$, as well as the validity is equivalent to the one of $\forall\mathsf{free}(\varphi)\varphi$. However, the proof is not as straightforward as for the classic semantics case.

Theorem 3. *For every* QLTL$_\mathsf{B}$ *formula* $\varphi = \wp\psi$, φ *is satisfiable if, and only if,* $\exists\mathsf{free}(\varphi)\varphi$ *is satisfiable. Moreover,* φ *is valid if, and only if,* $\forall\mathsf{free}(\varphi)\varphi$ *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 $F = \mathsf{free}(\varphi)$ such that $\pi \models_\mathtt{B} \varphi$, which in turns implies that there exists a behavioral Skolem function θ over $(\wp, F)$ such that $\hat{\theta}(\pi \Cup \pi') \models_\mathtt{C} \psi$ 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 $\hat{\theta'}(\pi') \models_\mathtt{LTL} \psi$ 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 $\hat{\theta}(\pi) \models_\mathtt{LTL} \psi$ for every $\pi \in (2^{\forall(\wp)\cup\{F\}})^\omega$. Observe that $\mathsf{Dep}^F_{\exists F\wp}(F) = \emptyset$ [5], 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 $\theta'(\pi_F \cup \pi) \models_\mathtt{LTL} \psi$ for every $\pi \in (2^{\forall(\wp)})^\omega$, from which we derive that $\pi_F \models_\mathtt{B} \wp\psi$, 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$_\mathsf{B}$ 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 $\varphi = \forall x \exists y(y \leftrightarrow \mathtt{G}x)$. Indeed, we have.

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

Lemma 1. *For every* QLTL$_\mathsf{B}$ *formula* φ *and every trace* π *over the set* $\mathsf{free}(\varphi)$ *of free variables, if* $\pi \models_\mathtt{B} \varphi$ *then* $\pi \models_\mathtt{C} \varphi$*. On the other hand, there exists a formula* φ *and a trace* π *such that* $\pi \models_\mathtt{C} \varphi$ *but not* $\pi \models_\mathtt{B} \varphi$*.*

7 QLTL$_B$ Satisfiability

There are three syntactic fragments for which QLTL and QLTL$_\mathsf{B}$ are equivalent. Precisely, the fragments $\Pi_0^{\text{QLTL}_\mathsf{B}}$, $\Sigma_0^{\text{QLTL}_\mathsf{B}}$, and $\Sigma_1^{\text{QLTL}_\mathsf{B}}$. Recall that $\Pi_0^{\text{QLTL}_\mathsf{B}}$ formulas are of the form $\forall X \varphi_{\text{LTL}}$, whereas $\Sigma_0^{\text{QLTL}_\mathsf{B}}$ formulas are of the form $\exists Y \varphi_{\text{LTL}}$. Finally, $\Sigma_1^{\text{QLTL}_\mathsf{B}}$ formulas are of the form $\exists Y \forall X \varphi_{\text{LTL}}$. 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$_\mathsf{B}$ *formula* $\varphi = \wp\psi$ *in the fragments* $\Pi_0^{\text{QLTL}_\mathsf{B}}$, $\Sigma_0^{\text{QLTL}_\mathsf{B}}$, *and* $\Sigma_1^{\text{QLTL}_\mathsf{B}}$ *and every trace* π*, it holds that* $\pi \models_\mathtt{B} \varphi$ *if, and only if,* $\pi \models_\mathtt{C} \varphi$*.*

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 $\varphi \in \Pi_0^{\text{QLTL}}$. Observe that $\exists(\wp) = \emptyset$ and so the only possible Skolem function θ returns the empty interpretation on every possible interpretation $\pi \uplus \pi' \in (2^{\mathsf{free}(\varphi) \cup \forall(\wp)})^\omega$. Such Skolem function is trivially behavioral and so we have that $\pi \models_\mathtt{S} \varphi$ implies $\pi \models_\mathtt{B} \varphi$.

For the case of $\varphi \in \Sigma_0^{\text{QLTL}} \cup \Sigma_1^{\text{QLTL}}$, assume that $\pi, \models_\mathtt{S} \varphi$ and let θ be a Skolem function such that $\theta(\pi \uplus \pi') \models_\mathtt{C} \varphi$ for every $\pi' \in (2^{\forall(\wp)})^\omega$. Observe that, for every $Y \in \exists(\wp)$, it holds that $\mathsf{Dep}_\wp = \emptyset$ and so the values of Y depend only on the free variables in φ. Now, consider the Skolem function θ' over $(\wp, \mathsf{free}(\varphi))$ defined such that as $\theta'(\pi') \doteq \theta(\pi'_{\restriction \forall(\wp) \uplus \pi})$. As θ is a Skolem function and $\mathsf{Dep}_\wp = \emptyset$, it holds that $\theta'(\pi')(Y) = \theta'(\pi'')(Y)$ for every $\pi', \pi'' \in (2^{\forall(\wp)})^\omega$ and so θ' is trivially behavioral. Moreover, from its definition, it holds that $\theta'(\pi \uplus \pi') \models_\mathtt{C} \psi$ for every $\pi' \in (2^{\forall(\wp)})^\omega$, which implies $\pi \models_\mathtt{B} \varphi$.

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

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

We now turn into solving satisfiability for QLTL$_\mathsf{B}$ formulas that are not in fragments $\Pi_0^{\text{QLTL}_\mathsf{B}}$, $\Sigma_0^{\text{QLTL}_\mathsf{B}}$, and $\Sigma_1^{\text{QLTL}_\mathsf{B}}$. 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 [31, 22, 38]. Details about this construction are available in the appendix. We have the following.

Theorem 5. *Satisfiability of* n-QLTL$_\mathsf{B}$ *is* $(n+1)$*-EXPTIME-complete.*

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 [21]. 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 [35]. 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 [30].

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 Conclusion

We introduced a behavioral variant of QLTL. Our variant, QLTL_B, is based on the following ingredients. First, it uses the syntax of QLTL. Secondly, it interprets the existential quantifications $\exists Y$ as functions from histories to the next value of Y, where the variables observed over the histories are controlled by the nesting of quantification. Third, satisfiability over this logic corresponds to advanced forms of reactive synthesis with partial observability.

Recently, independently of our work, QLTL has been at the base of a proposal that shares with us a strategic nature [8]. As witnessed by the complexity characterization of satisfiability in the two cases, respectively $(n+1)$-EXPTIME-complete, with n being the number of quantification blocks in our case, and 2-EXPTIME-complete in [8], our proposal looks at more sophisticated forms of strategies, with respect to partial observability over the histories. Deeper understanding on the relationship between the two approaches deserves further investigation.

Acknowledgements This work was supported by MUR under the PRIN programme, grant B87G22000450001 (PINPOINT), the ERC Advanced Grant WhiteMech (No. 834228), by the EU ICT-48 2020 project TAILOR (No. 952215), by the PRIN project RIPER (No. 20203FFYLK), the JPMorgan AI Faculty Research Award "Resilience-based Generalized Planning and Strategic Reasoning", and PNRR MUR project PE0000013-FAIR.

References

1. Abadi, M., Lamport, L., Wolper, P.: Realizable and unrealizable specifications of reactive systems. In: ICALP'89. LNCS, vol. 372, pp. 1–17. Springer (1989)
2. Alur, R., Henzinger, T., Kupferman, O.: Alternating-Time Temporal Logic. JACM **49**(5), 672–713 (2002)
3. Aminof, B., De Giacomo, G., Murano, A., Rubin, S.: Planning under LTL environment specifications. In: ICAPS. pp. 31–39. AAAI Press (2019)
4. Aminof, B., De Giacomo, G., Rubin, S.: Stochastic Fairness and Language-Theoretic Fairness in Planning in Nondeterministic Domains. In: ICAPS. pp. 20–28. AAAI Press (2020)
5. Bacchus, F., Kabanza, F.: Planning for Temporally Extended Goals. Ann. Math. Artif. Intell. **22**(1-2), 5–27 (1998)
6. Bacchus, F., Kabanza, F.: Using Temporal Logics to Express Search Control Knowledge for Planning. Artif. Intell. **116**(1-2), 123–191 (2000)
7. Barringer, H., Fisher, M., Gabbay, D.M., Gough, G., Owens, R.: METATEM: an introduction. Formal Aspects Comput. **7**(5), 533–549 (1995)
8. Bellier, D., Benerecetti, M., Monica, D.D., Mogavero, F.: Good-for-game QPTL: an alternating hodges semantics. ACM Trans. Comput. Log. **24**(1), 4:1–4:57 (2023)
9. Bertoli, P., Cimatti, A., Roveri, M.: Heuristic search + symbolic model checking = efficient conformant planning. In: IJCAI'01. pp. 467–472 (2001)
10. Calvanese, D., De Giacomo, G., Vardi, M.Y.: Reasoning about Actions and Planning in LTL Action Theories. In: KR'02. pp. 593–602 (2002)
11. Camacho, A., Bienvenu, M., McIlraith, S.A.: Towards a Unified View of AI Planning and Reactive Synthesis. In: ICAPS'19. pp. 58–67 (2019)
12. Cerrito, S., Mayer, M.: Bounded Model Search in Linear Temporal Logic and Its Application to Planning. In: TABLEAUX'98. LNCS, vol. 1397, pp. 124–140. Springer (1998)
13. Chatterjee, K., Henzinger, T.A.: Assume-Guarantee Synthesis. In: TACAS'07. LNCS, vol. 4424, pp. 261–275 (2007)
14. Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Environment Assumptions for Synthesis. In: CONCUR'08. pp. 147–161 (2008)
15. Chatterjee, K., Henzinger, T.A., Piterman, N.: Strategy logic. Inf. Comput. **208**(6), 677–693 (2010). https://doi.org/10.1016/j.ic.2009.07.004
16. Church, A.: Logic, arithmetics, and automata. In: Proc. Int. Congress of Mathematicians, 1962. pp. 23–35 (1963)
17. Cimatti, A., Giunchiglia, F., Giunchiglia, E., Traverso, P.: Planning via model checking: A decision procedure for *AR*. In: ECP'97. LNCS, vol. 1348, pp. 130–142 (1997)
18. Cimatti, A., Roveri, M.: Conformant Planning via Symbolic Model Checking. J. Artif. Intell. Res. **13**, 305–338 (2000)
19. Cimatti, A., Roveri, M., Traverso, P.: Strong planning in non-deterministic domains via model checking. In: AIPS. pp. 36–43. AAAI (1998)
20. Daniele, M., Traverso, P., Vardi, M.Y.: Strong cyclic planning revisited. In: ECP'99. LNCS, vol. 1809, pp. 35–48. Springer (1999)
21. De Giacomo, G., Vardi, M.: Automata-Theoretic Approach to Planning for Temporally Extended Goals. In: ECP. LNCS, vol. 1809, pp. 226–238. Springer (1999)
22. Finkbeiner, B., Schewe, S.: Uniform distributed synthesis. In: LICS'05. pp. 321–330 (2005)

23. Gabbay, D.M., Pnueli, A., Shelah, S., Stavi, J.: On the temporal basis of fairness. In: Abrahams, P.W., Lipton, R.J., Bourne, S.R. (eds.) POPL'80. pp. 163–173 (1980)
24. Gardy, P., Bouyer, P., Markey, N.: Dependences in Strategy Logic. In: STACS'18. LIPIcs, vol. 96, pp. 34:1–34:15 (2018)
25. Gardy, P., Bouyer, P., Markey, N.: Dependences in strategy logic. Theory Comput. Syst. **64**(3), 467–507 (2020)
26. Geffner, H., Bonet, B.: A Concise Introduction to Models and Methods for Automated Planning. Morgan & Claypool (2013)
27. Ghallab, M., Nau, D.S., Traverso, P.: Automated Planning - Theory and Practice. Elsevier, 1st edn. (2004), 635p
28. Goldman, R.P., Boddy, M.S.: Expressive Planning and Explicit Knowledge. In: Proc. of AIPS (1996)
29. Green, C.C.: Application of theorem proving to problem solving. In: IJCAI'69. pp. 219–240 (1969)
30. Kupferman, O., Vardi, M.Y.: Synthesis with Incomplete Informatio, pp. 109–127. Springer (2000)
31. Kupferman, O., Vardi, M.Y.: Synthesizing distributed systems. In: LICS'01. pp. 389–398 (2001)
32. Mogavero, F., Murano, A., Perelli, G., Vardi, M.: Reasoning about strategies: On the model-checking problem. ACM TOCL **15**(4), 34:1–34:47 (2014)
33. Nathanaël Fijalkow and Bastien Maubert and Aniello Murano and Sasha Rubin and Moshe Y. Vardi: Public and private affairs in strategic reasoning. In: Kern-Isberner, G., Lakemeyer, G., Meyer, T. (eds.) KR'22 (2022), https://proceedings.kr.org/2022/14/
34. Pnueli, A.: The temporal logic of programs. In: FOCS-77. pp. 46–57 (1977)
35. Pnueli, A., Rosner, R.: On the Synthesis of a Reactive Module. In: POPL. pp. 179–190. ACM (1989)
36. Pnueli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: FOCS'90. pp. 746–757 (1990)
37. Rintanen, J.: Complexity of Planning with Partial Observability. In: ICAPS'04. pp. 345–354 (2004)
38. Schewe, S.: Synthesis of distributed systems. Ph.D. thesis, Saarland University, Saarbrücken, Germany (2008)
39. Sistla, A., Vardi, M., Wolper, P.: The Complementation Problem for Büchi Automata with Applications to Temporal Logic. TCS **49**, 217–237 (1987)
40. Sistla, A.P.: Theoretical Issues in the Design and Verification of Distributed Systems. Ph.D. thesis (1985)
41. Thomas, W.: Languages, automata, and logic. In: Handbook of Formal Languages, Volume 3: Beyond Words, pp. 389–455. Springer (1997)