Physics-based Motion Planning with Temporal Logic Specifications

Muhayyuddin, Aliakbar Akbari, Jan Rosell

Institute of Industrial and Control Engineering (IOC), Universitat Politècnica de Catalunya (UPC) – Barcelona Tech, Spain.

Abstract: One of the main foci of robotics is nowadays centered in providing a great degree of autonomy to robots. A fundamental step in this direction is to give them the ability to plan in discrete and continuous spaces to find the required motions to complete a complex task. In this line, some recent approaches describe tasks with Linear Temporal Logic (LTL) and reason on discrete actions to guide sampling-based motion planning, with the aim of finding dynamically-feasible motions that satisfy the temporal-logic task specifications. The present paper proposes an LTL planning approach enhanced with the use of ontologies to describe and reason about the task, on the one hand, and that includes physics-based motion planning to allow the purposeful manipulation of objects, on the other hand. The proposal has been implemented and is illustrated with didactic examples with a mobile robot in simple scenarios where some of the goals are occupied with objects that must be removed in order to fulfill the task.

Keywords: Physics-based motion planning, sampling-based LTL planning, knowledge-based reasoning.

1. INTRODUCTION

The field of motion planning is evolving rapidly and one of the main directions is towards enabling robots to perform complex tasks in realistic environments. In this regard, on the one hand, it is evolving towards the simultaneous consideration of kinodynamic and physics-based constraints, like the physics-based motion planning approaches that allow to also consider the purposeful manipulation of objects. On the other hand, it is evolving towards the integration with task planning, like the LTL-based approaches that define multiple goal tasks in terms of temporal logic, such as *visit region A followed by region C and avoid region B*.

LTL-based motion planning is a hybrid approach of discrete (high-level) and continuous (low-level) planning, that computes the robot trajectories in such a way that they satisfy the temporal constraints, represented in terms of an LTL formula. To our best knowledge, all state-of-the-art LTL motion planers, such as Fainekos et al. (2005); Bhatia et al. (2010); Plaku (2012b); Lahijanian et al. (2012); Plaku et al. (2013); Edelkamp and Plaku (2014), evaluate the validity of the formula using model checking techniques (automaton construction) and in case of feasible formula, determine the collision-free trajectory that satisfy the formula. These planners neither analyze the formula against the capability of the robot, nor incorporate manipulation actions (i.e. they do not consider the dynamic interactions between rigid bodies). Therefore they are not able to compute the plan if, for instance, no collision free trajectory exists for moving between regions that the robot must visit, although with the removal of few objects a trajectory could be found.

⋆ This work was partially supported by the Spanish Government through the projects DPI2013-40882-P, DPI2014-57757-R and DPI2016-80077-R. Muhayyuddin is supported by the Generalitat de Catalunya through the grant FI-DGR 2014. A. Akbari is supported by the Spanish Gov. through the grant FPI 2015.

The straight extension of these approaches in order to handle manipulation actions is possible, but due to the high complexity of physics-based motion planning (large search space and highly constraint solution set) along with the incorporation of temporal constraints, may lead to a computationally non-tractable problem, raising the question about its decidability. Therefore, an efficient and powerful framework is required that has the capacity to handle both the temporal goals and the physics-based constraints, along with the purposeful manipulation of the objects if necessary. This paper tries to contribute in this line by enhancing the framework introduced in Muhayyuddin et al. (2015), that combined the use of ontologies with physics-based motion planning, by allowing now the consideration of temporal constraints.

Contributions. The main contributions of this paper are: *(1)* the integration of the Linear Temporal Logic within the framework of ontological physics-based motion planning, thus allowing the purposeful manipulation of objects, like the execution of push actions to clear regions possibly occupied by objects; *(2)* the proposal of an LTL feasibility evaluation and simplification process, that uses knowledge-based reasoning to evaluate whether the properties of the objects and the robot make the task described by the LTL formula feasible or not and, if required, simplifies the formula by skipping the non-valid propositions defined with disjunction relations.

The rest of the paper is structured as follows. Sec. 2 presents some relevant work related to physics-based motion planning and LTL-based motion planing. Sec. 3 describes the modeling of the world and problem statement. Then, Sec. 4 explains the framework, the reasoning process and the planning process. Implementation issues and some simulation results are explained in Sec. 5. Finally, the conclusions are presented in Sec. 6.

2. RELATED WORK

2.1 Physics-based Motion Planning

Motion planning problems focus on computing a collision-free trajectory from a given start state to the goal state in the configuration space. The configuration space $\mathcal{C}$ represents the set of all possible configurations of the robot; it is divided into $\mathcal{C}_{free}$ (free regions) and $\mathcal{C}_{obs}$ (forbidden regions). To take into account differential (dynamic) constraints the planning is performed in a higher dimensional state space $\mathcal{S}$ that records the system dynamics. For any configuration $q \in \mathcal{C}$ the state of the system is represented as $s = (q, \dot{q})$. Planning will be performed in $\mathcal{S}$ in a similar way as it is done in $\mathcal{C}$ for pure geometric problems.

Kinodynamic motion planning is centered in computing the collision-free trajectories that satisfy the kinodynamic constraints (such as joint limits, bounds on the forces and acceleration). Sampling-based motion planners (Tsianos et al., 2007), particularly those based on tree data structures, are well suited for this purpose because the state propagation used to incrementally grow the data structures can easily take into account the constraints. Moreover, if a dynamic engine is used as state propagator then physics-based constraints (gravity and friction) can also be easily incorporated (Şucan and Kavraki, 2012), (Plaku, 2012a). Therefore, physics-based motion planning can be considered as an evolved form of kinodynamic planning in which the robot motions also satisfy physics-based constraints. This, moreover, allows to evaluate the dynamic interaction between rigid bodies and, besides considering only collision-free trajectories, allows the consideration of manipulation actions (such as the push action), thus broadening the range of tasks that can be solved.

The physics-based motion planners rely on sampling-based kinodynamic motion planners, such as Rapidly-Exploring Random Trees (RRT) (LaValle and Kuffner, 2001), Kinodynamic Motion Planning by Interior-Exterior Cell Exploration (KPIECE) (Şucan and Kavraki, 2012), Synergistic Combination of Layers of Planning (SyCLoP) (Plaku et al., 2010), for sampling the states and constructing the solution path, while the state propagation is performed using a dynamic engine such as Open Dynamic Engine (ODE) (Russell, 2007). The large search space and the evaluation of the dynamical interactions make physics-based motion planning computationally intensive. A few approaches have been proposed that try to overcome these issues, such as the work of Zickler and Veloso (2009) that proposed a nondeterministic tactic based on a finite state machine to guide the motion planner, along with the use of action skills to control the sampling. In a similar direction, the ontological physics-based motion planning approach of Muhayyuddin et al. (2015) performs a knowledge-based reasoning process to determine the way of manipulating objects, thus reducing the planning search space. This knowledge-based framework can be used together with any sampling-based kinodynamic motion planner (such as RRT, KPIECE, SyCLoP), and ODE is set as state propagator. Moreover, this framework has also been used for computing the motion plan and dynamic cost in integrated task and motion planning approaches such as Akbari et al. (2015, 2016). The present proposal extends this approach for temporal goals described by an LTL formula.

2.2 LTL-based Motion Planning

LTL motion planning is a hybrid approach that provides a framework to describe complex motion planning tasks in terms of temporal goals, and that plans in discrete and continuous spaces. The planning is performed in three steps (1) *Workspace decomposition:* decomposes the robot workspace (using for example a triangular decomposition); (2) *High level planning:* constructs the discrete plan over the product space of the decomposed workspace and the automaton (that is constructed to check the LTL formula) in such a way that the discrete plan satisfies the LTL formula ϕ; (3) *Low level planning:* implements the high level plan at low level in such a way that it also satisfies ϕ.

LTL-based motion planning approaches are broadly divided into two main categories: controller-based and sampling-based LTL motion planers. The former compute the discrete plan over the decomposed workspace and then the controller looks for the dynamically-feasible and collision-free trajectory for each action (Fainekos et al., 2009). The latter consider the integration of the task and motion planning steps, proposing a probabilistic search over the hybrid space of discrete and continuous components (Bhatia et al., 2010, 2011; Maly et al., 2013; Plaku et al., 2013; He et al., 2015). The discrete component is represented as the product space of decomposed workspace and the automaton that satisfies the LTL formula ϕ, whereas the continuous layer consists of a sampling-based dynamic motion planner that is guided by the discrete layer. All these approaches always seek for a collision-free trajectory. Although some approaches such as (McMahon and Plaku, 2014) incorporate a dynamic engine within an LTL framework, it is used only to consider the robot dynamics and the physics-based constraints, i.e. no dynamic interactions between rigid bodies are considered while planning.

3. PRELIMINARIES

3.1 Modeling

Consider the world is composed of a set of rigid bodies $\mathcal{B}$, that are categorized into fixed and movable. The former remains fixed throughout the planning process and is represented as $\mathcal{B}_{\text{fixed}}$, whereas the latter can be moved (pushed) by the robot. The movable bodies are further divided into:

- Freely-movable bodies $\mathcal{B}_{\text{free}}$: Bodies that can be manipulated freely from any direction.
- Constraint-oriented movable bodies $\mathcal{B}_{\text{co}}$: Bodies that must be manipulated from certain directions, such as car-like bodies that can only be pushed in the forward or backward directions.

The manipulation constraints of a $\mathcal{B}_{\text{co}}$ are modeled by defining some part of the body that the robot is allowed to touch, and an associated region, called manipulation region. (*mRegion*), where the robot must be located in order to interact with the body. All bodies in the environments can be represented as; $\mathcal{B} = \mathcal{B}_{\text{free}} \cup \mathcal{B}_{\text{co}} \cup \mathcal{B}_{\text{fixed}}$.

To store the above stated information and to update it while planning, the knowledge is represented in two levels, the abstract knowledge $\mathcal{K}$ and the instantiated knowledge κ. The abstract knowledge is represented using ontologies encoded with the Web Ontology Language (OWL) (Antoniou and van Harmelen, 2003). It contains the type of the objects (such as fixed and manipulatable), their properties (such as masses and friction coefficients), associated manipulation constraints, kinodynamic properties of the robot (such as joint limits, bounds on the forces, torques and velocities), and the LTL operators. The abstract knowledge $\mathcal{K}$ remains fixed throughout the planning process. The instantiated knowledge κ is the dynamic knowledge, inferred from $\mathcal{K}$ through the reasoning process, and updated continuously at each time step. It contains the manipulation constraints that are valid at each particular instance of time.

Let $\mathcal{X}$ be the state space of all the bodies in the environment. At any time t a state $x \in \mathcal{X}$ is represented as $x(t) = \{s_1 \dots s_k\}$ where s_i represents the position and orientation of the i-th object in the environment. The instantiated knowledge κ_t^x is associated to state x.

3.2 Robot Model

Consider a mobile robot $\mathcal{R}$, and let $\mathcal{S}$ be its state space containing all possible states of the robot. A state $s \in \mathcal{S}$ is represented as $s = \{p, o, v, w\}$ where p, o, v, and w are the position, orientation, linear velocity and angular velocity respectively. The instantiated knowledge κ_t^s is associated to s. The state of the environment can be represented as $E = \mathcal{X} \times \mathcal{S}$.

A trajectory of the robot is defined by the robot dynamics in the results of control inputs, that are applied for a small time duration Δt. It can be written as $s_{\text{new}} = \text{PROPAGATOR}(s, u, \Delta t)$, where $u \in \mathcal{U}$ is a control input from the control space $\mathcal{U}$ containing the set of all possible control inputs that can be applied to the robot. The PROPAGATOR will generate a trajectory between state s and s_{new}. An entire trajectory (*Traj*) of the robot is obtained by applying the control inputs (starting from start state) repeatedly for small time durations. During the execution of the motion, if the robot interact with $\mathcal{B}_{\text{free}}$ or $\mathcal{B}_{\text{co}}$ the resulting motion will change the state of the bodies. It implies that control inputs are responsible for updating the state of the robot as well as the state of the bodies. Generally the transition function PROPAGATOR can be written as PROPAGATOR: $E_i \times \mathcal{U} \rightarrow E_{i+1}$. The instantiated knowledge for E can be defined as $\kappa = \kappa^s \cup \kappa^x$. validity of newly generated state is evaluated by a function VALIDITYCHECKER: $E \times \kappa \rightarrow \{\top, \perp\}$. It returns $\top$ iff the newly generated state satisfies all the constraints imposed by κ. For each new state of E, κ is updated by the inference process INFERENCE: $E_{i+1} \times \kappa_i \rightarrow \kappa_{i+1}$.

3.3 Linear Temporal Logic

Linear temporal logic is a formalism used to specify tasks by combining propositions with logical and temporal operators. The combination is called an LTL formula ϕ (Clarke et al., 1999), i.e. an LTL formula ϕ is defined by integrating propositions with the logic operators *negation* ($\neg$), *conjunction* ($\wedge$), *disjunction* ($\vee$), *equivalence* ($\Leftrightarrow$), and *implication* ($\Rightarrow$) along with the temporal operators *next* ($\bigcirc$), *always* ($\Box$), *until* ($\sqcup$), and *eventually* ($\Diamond$).

Let Π be the set of atomic propositions, $\Pi = \{\pi_1, \dots, \pi_n\}$, each π_i representing a statement such as *"robot is in region P_i"*. Every $\pi_i \in \Pi$ is a formula, and if ϕ and ψ are formulas, then new formulas can be defined using the following grammar:

$$\neg\phi,\ \phi \wedge \psi,\ \phi \vee \psi,\ \Diamond\phi,\ \phi \sqcup \psi,\ \phi \bigcirc \psi$$

As an example, the formula to visit the regions $P_1,\ P_2,\ P_3$ in an ordered way can be represented as $\phi = \Diamond(\pi_1 \wedge \Diamond(\pi_2 \wedge \Diamond(\pi_3)))$

The semantics of LTL formula are defined over infinite traces of a system. Let $\sigma = \tau_0, \tau_1, \dots \tau_\infty$ represent the infinite trace with $\tau_i \in 2^\Pi$, and $\sigma \models \phi$ represent that σ satisfies ϕ. Then, $\sigma \models \phi$ iff there exists a finite prefix $\sigma_i = \tau_0, \tau_1, \dots, \tau_{i-1}$ of σ that satisfies ϕ.

Syntactically co-safe formulas are a special class of LTL formulas. When written in positive normal form (i.e. when the negation operator occurs only in front of atomic propositions), they only contain the eventually, next and until operators. They can be interpreted over a finite trace and their validity can be checked using nondeterministic finite automata (Kupferman and Vardi, 2001).This is the type of formulas used when the focus is in motion planning problems over a finite time horizon.

3.4 LTL Semantics for Motion Trajectories

The robot workspace $\mathcal{W}$ contains different types of rigid bodies and a set of propositional regions $P = \{P_1 \dots P_n\}$ corresponding to the propositions $\{\pi_1 \dots \pi_n\}$. The part of the workspace that is accessible by the robot is represented as $\mathcal{W}_{\text{acc}} = \mathcal{W} \setminus \mathcal{W}_{\text{fixed}}$, where $\mathcal{W}_{\text{fixed}}$ is the part of the workspace occupied by $\mathcal{B}_{\text{fixed}}$. Propositional regions are associated with the accessible part of the workspace, i.e. $P \in \mathcal{W}_{\text{acc}}$. A special proposition π_0 is associated with a propositional region P_0 defined as $P_0 = \mathcal{W}_{\text{acc}} \setminus \cup_{i=1}^n P_i$. A function $\mathcal{G} : \mathcal{W}_{\text{acc}} \rightarrow \Pi$ maps each point of the workspace over the corresponding propositional region.

The discrete trace of a trajectory *Traj* is defined as the sequence of propositional regions that are traversed by *Traj* and is represented as $tr(\textit{Traj})$. A propositional region P_i is said to be traversed iff $\mathcal{G}(\textit{Traj}(t)) = \pi_i$ for some $0 \leq t \leq T$. A motion trajectory *Traj* satisfies ϕ iff $tr(\textit{Traj}) \models \phi$.

3.5 Problem Statement

Let a motion planning problem for a robot $\mathcal{R}$ be considered as the tuple $\langle E_{\text{init}}, \mathcal{K}, \Pi, \phi \rangle$. The goal is expressed in terms of an LTL formula ϕ, defined over a set of atomic propositions Π that describe regions of the workspace that the robot must either visit or avoid. The problem is to evaluate (by taking into account E_{init} and $\mathcal{K}$) whether the robot can satisfy the formula, and whether it can be simplified. If the formula (or possibly simplified formula) is feasible then the problem is to find a sequence of control inputs such that the resulting robot trajectory *Traj* satisfies $tr(\textit{Traj}) \models \phi$, is dynamically feasible, avoids the collision with fixed bodies and, if necessary, pushes movable objects away for clearing the regions.

4. PHYSIC-BASED LTL MOTION PLANNING

4.1 Framework

To solve the above stated problem, a physics-based LTL motion planning approach is proposed that makes use of ontologies and high-level reasoning, and that considers physics-based motion propagation. The schematic representation of the solution framework is depicted in Fig. 1. It consists of two main modules: the knowledge-based reasoning engine and the physics-based LTL planner. The former is responsible for defining the manipulation constraints, the feasibility evaluation and the

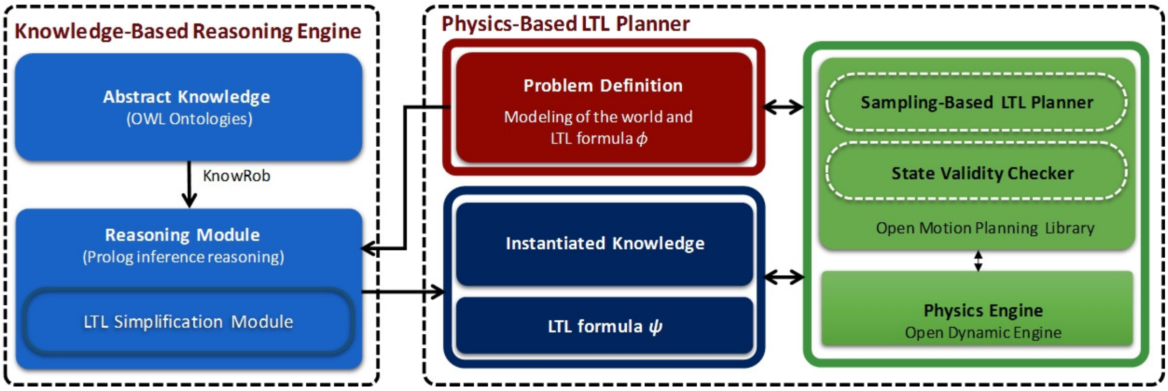

Fig. 1. Framework for Physics-based LTL motion planning.

possible simplification of the LTL formula, whereas the latter computes the motion plan that satisfies the temporal goals.

The knowledge-based reasoning engine contains the abstract knowledge $\mathcal{K}$, and performs a prolog-based reasoning over $\mathcal{K}$ to define the types of the rigid bodies and the associated manipulation constraints (encoded in the instantiated knowledge) using the kinematic and dynamic properties of the robot and the bodies. Furthermore, it is responsible for the feasibility evaluation and the possible simplification of the LTL formula (explained in Sec. 4.2). It is important to note that the feasibility evaluation and simplification is different from the validity checking of the LTL formulas that is performed using model-checking techniques.

The physics-based LTL planner stores the problem definition (the world modeling along with the initial LTL formula ϕ that defines the temporal goals), the instantiated knowledge κ and the LTL formula ψ inferred by the reasoning module (ψ contains a simplified version of ϕ, when possible). The automaton will be constructed for ψ, unlike other sampling-based LTL planners that always plan for ϕ. The sampling-based LTL motion planner is responsible for generating the discrete plan (computed over the product space of the decomposed workspace and generated automaton for ψ) and its execution at low-level (continuous motion planning level) to determine the control sequence in such a way that the resultant trajectory satisfies ψ. The state propagator makes use of the physics engine for the propagation, and the newly generated states are evaluated by the state validity checker. Different from standard sampling-based LTL planners, the proposed validity checker takes into account the current state of the environment and evaluates it based on the instantiated knowledge that is valid for that particular state.

4.2 Reasoning Process

The aim of knowledge-based reasoning is to provide autonomy to the robots for performing complex tasks. We use the reasoning process, on one hand, to generate the instantiated knowledge κ and, on the other hand, to evaluate the feasibility of the LTL formula and its potential simplification.

For the generation of κ a prolog-based reasoning is employed that reads the abstract knowledge in order to classify the objects into different types, together with their manipulation constraints. This process is performed by evaluating physical properties of the objects and the kinodynamic properties of the robot in a similar way as performed in Muhayyuddin et al. (2015); Gillani et al. (2016). At each instant of time κ is updated using the INFERENCE function that takes into account the previous state of κ and the current state of the environment and generates the new state of κ (INFERENCE: $E_{i+1} \times \kappa_i \rightarrow \kappa_{i+1}$). For instance, if after a propagation step one manipulation region is occupied with another object, the motion constraints of the first object are updated accordingly to the new situation.

Algorithm 1 Simplify

Input: List L with nonvalid proposition(s), LTL formula ϕ
Output: LTL formula ϕ

if $op(L) \ni \vee$ **then**
    return $\phi \backslash L$
else
    if $parent(L) = \phi$ **then**
        return NULL
    else
        Simplify($parent(L), \phi$)
    end if
end if

Algorithm 2 Evaluate

Input: LTL formula ϕ, Set of propositions Π, Knowledge $\mathcal{K}$
Output: ϕ or NULL

$\mathcal{M}_p \leftarrow \mathcal{F}(\mathcal{K}, \Pi)$
if $\mathcal{M}_p =$ NULL **then**
    return ϕ
end if
for all $\pi \in \mathcal{M}_p$ **do**
    $\phi \leftarrow$Simplify($\pi.L, \phi$)
    if $\phi =$ NULL **then**
        return NULL
    end if
end for
return ϕ

To evaluate the feasibility and the possible simplification of the LTL formula, the reasoning process is done as follows:

- Let an LTL formula ϕ be defined over a set of propositions $\Pi = \{\pi_1 \dots \pi_n\}$, where each $\pi_i \in \Pi$ is associated with a region P_i of the workspace (that the robot should visit or avoid) called *propositional region*. A proposition is considered nonvalid if the associated propositional region is not accessible by the robot and valid otherwise. Let $\mathcal{M}_p$ be the list of nonvalid propositions and $\mathcal{F}$ be the function that computes them, i.e. $\mathcal{F} : \mathcal{K} \times \Pi \Rightarrow \mathcal{M}_p$. If $\mathcal{M}_p = \emptyset$, the formula is feasible and does not require simplification.

Algorithm 3 Physics-based LTL Motion Planning

Input: Initial state E_{init}, Π, LTL formula ϕ, Threshold T_{max}
Output: A continuous path that satisfies ϕ.
1: $\mathcal{K} \leftarrow$ OntologyFormulation(E_{init})
2: $\kappa_0 \leftarrow$InstantiatedKnowledgeInference($\mathcal{K}$)
3: $\psi \leftarrow$ Evaluate($\mathcal{K}, \Pi, \phi$)
4: **if** ψ = NULL **then**
5:    **return** NULL
6: **else**
7:    $\mathcal{T} \leftarrow$InitializeTree(E_{init})
8:    $\mathcal{A}_\psi \leftarrow$ComputeAutomaton($\psi$)
9:    $\mathcal{D} \leftarrow$ ComputeDecomposition() ; $j = 0$
10:    **while** $t < T_{max}$ **do**
11:       $\rho \leftarrow$ DiscretePlanning($\mathcal{A}_\psi$,$\mathcal{D}$)
12:       $v \leftarrow$ SelectHighLevelState(ρ)
13:       $\{u, n\} \leftarrow$ SampleControlAndSteps(v)
14:       **for** $i = 0$ **to** n **do**
15:          $\boldsymbol{E}_{\text{new}} \leftarrow$ PROPAGATOR(E, u,Δt)
16:          **if** ! VALIDITYCHECKER(E_{new},κ_j) **then**
17:             Break
18:          **else**
19:             $\kappa_{j+1} \leftarrow$ INFERENCE(E_{new}, κ_j); $j = j + 1$
20:             $v_{new} \leftarrow$UpdateHighLevelState(v)
21:             $\mathcal{T} \leftarrow$ UpdateTree($E_{new}, u, \Delta t$)
22:             $z \leftarrow$GetAutomatonState(v_{new})
23:             **if** $z \in$ Accepting state of $\mathcal{A}_\psi$ **then**
24:                **return** *Traj*$\leftarrow$RetrieveTrajectory($\mathcal{T}$)
25:             **end if**
26:          **end if**
27:       **end for**
28:    **end while**
29:    **return** NULL
30: **end if**

- Let a formula be considered a list L, which can be either an atomic list (defined by a single proposition), or a compound list (defined by the composition of lists using temporal and logic operators). Then, $parent(L)$ returns the parent list of a list, $op(L)$ returns the set of prefix and postfix operators of L within $parent(L)$, and $\pi.L$ represents the innermost list containing the proposition π (i.e. $\pi.L = \{\pi\}$).
- Let Simplify(L, ϕ) be a recursive function that verifies if L contains disjunction operators and if so returns the formula ϕ without L, as shown in Algorithm 1. Then, Algorithm 2 shows the procedure Evaluate(ϕ) that checks the feasibility of an LTL formula by using function Simplify(L, ϕ).

As an example, consider the formula $\phi = \Diamond\pi_1 \vee (\Diamond\pi_2 \wedge (\Diamond\pi_3))$ where the proposition π_3 is nonvalid. The list associated to π_3 is $\mathcal{L}_{(3,1)}$, as shown in Fig. 2. The recursive Simplify is initially called for $\mathcal{L}_{(3,1)}$, then for $\mathcal{L}_{(2,2)}$ and $\mathcal{L}_{(1,2)}$, when the $\vee$ operator is found. At this moment the formula is simplified by deleting $\mathcal{L}_{(1,2)}$ and the simplified formula results $\psi = \Diamond\pi_1$.

The proposed reasoning process works for the syntactically co-safe LTL formulas. The Open Motion Planning Library (OMPL) (Şucan et al., 2012), a C++ based tool for sampling-based motion planning, provides the implementation of the sampling-based LTL motion planner presented in Bhatia et al. (2010). It supports the temporal goals defined using syntactically co-safe LTL formulas such as $\phi = \Diamond\pi_1 \wedge \cdots \wedge \Diamond$ (visit all the regions $\pi_1 \ldots \pi_n$ in any order), $\phi = \Diamond(\pi_1 \wedge \Diamond(\pi_2 \wedge \Diamond(\ldots \wedge \Diamond\pi_n)))$ (visit all the regions in an ordered way), $\phi = \Diamond\pi_1 \vee \cdots \vee \Diamond\pi_n$ (visit any of the region from $\pi_1 \ldots \pi_n$), and $\phi = (\Diamond\pi_1 \vee \Diamond\pi_2) \wedge \neg\pi_3$ (visit π_1 or π_2 and not visit π_3).

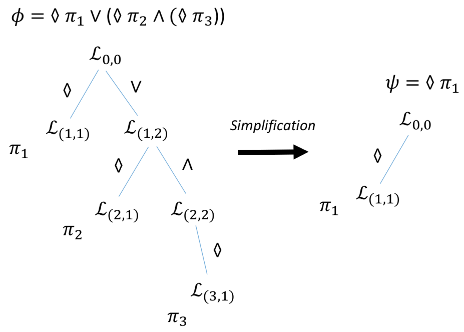

Fig. 2. Example of the simplification process, where ϕ and ψ are the actual and the simplified formulas respectively. Each $\mathcal{L}_{(i,j)}$ is a list with i and j representing the depth and the order in the parent list, respectively.

The reasoning process works over the similar types of formula with similar grammar.

4.3 Planning Process

The planning process is explained in Algorithm 3, that takes as inputs the initial state, the set of propositions Π, the temporal goal (defined in terms of an LTL formula ϕ over Π), and the maximum allowed planning time T_{max}. As output it returns a continuous path (as a sequence of controls and durations) that satisfies ϕ.

The *OntologyFormulation* function defines the abstract knowledge $\mathcal{K}$ about the world by defining the types of the objects (such as fixed or manipulatable), and their manipulation constraints in terms of *mRegions* (Sec.3.1). The *InstantiatedKnowledgeInference* fills the initial state of the instantiated knowledge as explained in Sec. 4.2.To determine the feasibility of ϕ, the *Evaluate* function computes the feasibility and performs the possible simplification (if required) as explained in Algorithms 1 and 2. The function *InitializeTree* sets the initial state of the tree as the initial state of the environment.

Lines: (8-13) refer to the general steps of the sampling-based LTL motion planning, as done in Bhatia et al. (2010), for the high level planning and for the updating of the states (both at low- and high- levels). *ComputeAutomaton* function computes the Automaton $\mathcal{A}_\psi$ for the formula ψ, *ComputeDecomposition* performs the triangular decomposition of the workspace not occupied by fixed obstacles, in a way that preserves the propositional regions. As a difference with Bhatia et al. (2010) we only exclude from the decomposition the workspace occupied by fixed obstacles, i.e. the non-fixed bodies are simply ignored while decomposing the workspace. Function *DiscretePlanning* constructs the discrete plan over the product space of the decomposition and $\mathcal{A}_\psi$, and *SelectHighLevelState* selects the high-level state to be explored. At low level, *SampleControlAndSteps* function samples the controls (that could be a vector of applied forces, joints torques, or velocities), and the number of steps (that refer to the number of times that the sampled controls will repeatedly applied for a duration Δt).

The function PROPAGATOR applies the sampled controls for Δt time on the robot and generates new state of the environment E_{new} using the dynamics engine that allows to handle all the kinodynamic and physics-based constraints. VALIDITYCHECKER evaluates the newly generated state of the environment, based on the instantiated knowledge κ. E_{new} will be accepted if

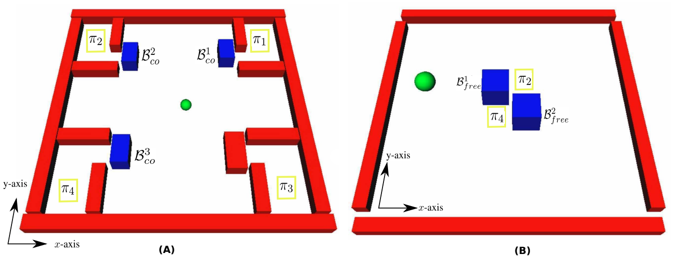

Fig. 3. Example scenarios A: the goal is to visit the propositional regions $\pi_1, \cdots, \pi_4$, being the access to them obstructed by the blue boxes. B: visit all propositional regions in an ordered way, being two of them occupied by bodies. Video: https://sir.upc.edu/projects/kautham/videos/IFAC2017.mp4

it satisfies all the constraints (such as temporal constraints, kinodynamic and physics-based constraints) that are imposed by κ and discarded otherwise.

The INFERENCE function updates the instantiated knowledge with the manipulation constraints that are valid for E_{new}. *UpdateHighLevelState* updates the high-level state based on the result of the low-level state and *UpdateTree* updates the tree-data structure. The *GetAutomatonState* function determines the state of the automaton and if it is the accepting state of $\mathcal{A}_\psi$, the *RetrieveTrajectory* function returns the continuous trajectory *Traj* such that $tr(Traj) \models \phi$.

5. RESULTS AND DISCUSSION

The physics-based LTL motion planning framework (depicted in Fig. 1) is implemented within *The Kautham Project* (Rosell et al., 2014) that is a motion planning framework that mainly uses planers from the OMPL (Şucan et al., 2012). For the current proposal a variant of the LTL has been implemented, and the Open Dynamic Engine has been used for the state propagation. The abstract knowledge $\mathcal{K}$ is implemented (in term of OWL ontologies) using the *Protégé* editor. Instantiated knowledge is defined by applying the prolog-based reasoning process, it uses predicates (functions) defined in Knowrob (Tenorth and Beetz, 2009), which is a knowledge processing framework for robots, to access the information from $\mathcal{K}$. The communication between the modules is performed using ROS (Quigley et al., 2009).

The simulation setup consists of a robot (green sphere), constraint-oriented movable bodies (blue cubes), and fixed bodies (red walls). There are two scenarios presented in Fig. 3. In the first scenario, it is assumed that all the propositional regions are surrounded by objects (fixed or movable) such that no collision-free trajectory exists to visit each of these regions. It represents different rooms that robot has to visit but, in order to enter each room, the robot has to interact with the body blocking the entrance. The propositional regions $\{\pi_1 \ldots \pi_4\}$, are shown as yellow rectangles. The temporal goal is defined by the LTL formula $\phi = \diamondsuit(\pi_1 \wedge \diamondsuit(\pi_2 \wedge \diamondsuit(\pi_3 \vee \pi_4)))$ that is: visit π_1, π_2 and then π_3 or π_4 The region associated with π_3 is surrounded with $\mathcal{B}_{\text{fixed}}$ and therefore, the reasoning process marks π_3 as invalid (it is not accessible by the robot). Since it has a disjunction relation with the other propositions, the simplification process will simplify the formula to $\psi = \diamondsuit(\pi_1 \wedge \diamondsuit(\pi_2 \wedge \diamondsuit(\pi_4)))$. Since the length of the first body is greater than the entrance, at its current location its manipulation region along the x-axis of the world frame is occupied with the walls. Therefore, the reasoning process will change the status of the body from freely-movable to constraint-oriented movable, and only allow the robot to push it along the y-axis. If after pushing the body, all the manipulation regions become free, the INFERENCE function will change the type of $\mathcal{B}^1_{co}$ to $\mathcal{B}^1_{\text{free}}$. The similar process is applied for $\mathcal{B}^2_{\text{co}}$ and $\mathcal{B}^3_{\text{co}}$.

The temporal goal for the second scenario is described as $\phi = \diamondsuit(\pi_1 \wedge \diamondsuit(\pi_3 \wedge \diamondsuit(\pi_2 \wedge \diamondsuit(\pi_4))))$. That is, visit π_1, π_3, π_2 and π_4 consecutively. The propositional regions associated to π_1 and π_3 are occupied by $\mathcal{B}^1_{free}$ and $\mathcal{B}^2_{free}$ respectively. Therefore, in order to visit π_1 (without prior being on π_2 or π_4), the robot must push $\mathcal{B}^1_{free}$ along the x-axis or $-y$-axis of the world frame. If it pushes $\mathcal{B}^1_{free}$ along the x-axis then it ends occupying the manipulation region of $\mathcal{B}^2_{free}$ that is along y-axis and hence the reasoning process will deactivate the *mRegion* along the $-y$-axis and change the status of the body to constraint-oriented movable. The same reasoning process is repeated for the second body, i.e. The INFERENCE function will update the types of these bodies to $\mathcal{B}^1_{co}$ and $\mathcal{B}^2_{co}$. To visit π_3, $\mathcal{B}^2_{co}$ can only be pushed along the $-x$-axis. After visiting π_3 the types of the bodies will be restored to $\mathcal{B}_{\text{free}}$ and the task can continue.

These two examples show that the proposed approach is able, on the one hand, to deal with movable objects that may be obstructing the solution path (changing if necessary the way the robot has to interact with them) and, on the other hand, is able to simplify a formula if part of it is non-feasible.

We tested both scenarios with and without instantiated knowledge. The simulation was performed on an Intel Core i7-4500U 1.80GHz CPU with 16 GB memory. For the first scene, the success rate of simple physics-based LTL planner was 30% for 10 runs (maximum allowed time was 300 seconds) whereas the success rate of the proposed approach was 80%. The simple physics-based LTL approach has an average planning time of 230 seconds. In contrast, the proposed approach computes the solution in 46.4 seconds (average of 10 runs).

For the second scenario, the success rate of the proposed approach and the simple physics-based approach were 100%. But, in the case of the proposed approach the quality of the solution was better, it avoids the unnecessary interactions between the robot and the objects and move the objects only when it is necessary. Regarding planning time, the proposed approach computes the solution in 2.1 seconds (average of 10 runs) and simple physics-based planning approach takes 23.8 seconds.

6. CONCLUSIONS

This paper has proposed the integration of LTL planning within the framework of ontological physics-based motion planning in order to provide robustness and autonomy for handling complex temporal goals in a realistic way. Moreover, a simplification process of the LTL formula is proposed, according to the validity or not of the goals to be satisfied and the logical operators involved. The proposed approach has been validated using simulation examples in which some of the propositional regions are occupied with objects or the way to the propositional region is blocked with objects that the robot has to push away, if possible, in order to visit the regions. The results shows that the integration of knowledge makes the planner more efficient and enhance the quality of the solution.

REFERENCES

Akbari, A., Muhayyuddin, and Rosell, J. (2015). Task and motion planning using physics-based reasoning. In *IEEE 20th Int. Conf. on Emerging Technologies Factory Automation (ETFA)*, 1–7.

Akbari, A., Muhayyuddin, and Rosell, J. (2016). Task planning using physics-based heuristics on manipulation actions. In *IEEE 21st Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, 1–8.

Antoniou, G. and van Harmelen, F. (2003). Web Ontology Language: OWL. In S. Staab and R. Studer (eds.), *Handbook on Ontologies in Information Systems*, 67–92. Springer-Verlag.

Bhatia, A., Kavraki, L.E., and Vardi, M.Y. (2010). Sampling-based motion planning with temporal goals. In *Robotics and Automation, IEEE Int. Conf. on*, 2689–2696.

Bhatia, A., Maly, M.R., Kavraki, L.E., and Vardi, M.Y. (2011). Motion planning with complex goals. *Robotics & Automation Magazine, IEEE*, 18(3), 55–64.

Clarke, E.M., Grumberg, O., and Peled, D. (1999). *Model checking*. MIT press.

Edelkamp, S. and Plaku, E. (2014). Multi-goal motion planning with physics-based game engines. In *Computational Intelligence and Games, IEEE Conf. on*, 1–8.

Fainekos, G.E., Girard, A., Kress-Gazit, H., and Pappas, G.J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, 45(2), 343–352.

Fainekos, G.E., Kress-Gazit, H., and Pappas, G.J. (2005). Temporal logic motion planning for mobile robots. In *Robotics and Automation, Proc. of the IEEE Int. Conf. on*, 2020–2025.

Gillani, M., Akbari, A., and Rosell, J. (2016). Physics-based motion planning: Evaluation criteria and benchmarking. In *Robot 2015: Second Iberian Robotics Conference*, 43–55. Springer.

He, K., Lahijanian, M., Kavraki, L.E., and Vardi, M.Y. (2015). Towards manipulation planning with temporal logic specifications. In *Robotics and Automation, IEEE Int. Conf. on*, 346–352.

Kupferman, O. and Vardi, M.Y. (2001). Model checking of safety properties. *Formal Methods in System Design*, 19(3), 291–314.

Lahijanian, M., Andersson, S., and Belta, C. (2012). Temporal logic motion planning and control with probabilistic satisfaction guarantees. *Robotics, IEEE Trans. on*, 28(2), 396–409.

LaValle, S.M. and Kuffner, J.J. (2001). Randomized kinodynamic planning. *The Int. J. of Robotics Research*, 20(5), 378–400.

Maly, M.R., Lahijanian, M., Kavraki, L.E., Kress-Gazit, H., and Vardi, M.Y. (2013). Iterative temporal motion planning for hybrid systems in partially unknown environments. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, 353–362. ACM.

McMahon, J. and Plaku, E. (2014). Sampling-based tree search with discrete abstractions for motion planning with dynamics and temporal logic. In *Intelligent Robots and Systems, IEEE/RSJ Int. Conf. on*, 3726–3733.

Muhayyuddin, Akbari, A., and Rosell, J. (2015). Ontological physics-based motion planning for manipulation. In *Emerging Technologies Factory Automation, IEEE 20th Conf. on*, 1–7.

Plaku, E., Kavraki, L., and Vardi, M. (2010). Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Tran. on Robotics*, 26(3), 469–482.

Plaku, E. (2012a). Motion planning with discrete abstractions and physics-based game engines. In *Proc. of the Int. Conf. on Motion in Games*, 290–301. Springer.

Plaku, E. (2012b). Planning in discrete and continuous spaces: From ltl tasks to robot motions. In *Advances in Autonomous Robotics*, 331–342. Springer.

Plaku, E., Kavraki, L.E., and Vardi, M.Y. (2013). Falsification of ltl safety properties in hybrid systems. *Int. J. on Software Tools for Technology Transfer*, 15(4), 305–320.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A.Y. (2009). ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3, 5.

Rosell, J., Pérez, A., Aliakbar, A., Muhayyuddin, Palomo, L., and García, N. (2014). The kautham project: A teaching and research tool for robot motion planning. In *Emerging Technologies and Factory Automation, Proc. of the IEEE Int. Conf. on*.

Russell, S. (2007). Open Dynamic Engine. http://www.ode.org/.

Şucan, I.A. and Kavraki, L.E. (2012). A sampling-based tree planner for systems with complex dynamics. *IEEE Trans. on Robotics*, 28(1), 116–131.

Şucan, I.A., Moll, M., and Kavraki, L.E. (2012). The open motion planning library. *Robotics & Automation Magazine, IEEE*, 19(4), 72–82.

Tenorth, M. and Beetz, M. (2009). Knowrob knowledge processing for autonomous personal robots. In *Intelligent Robots and Systems, Proc. of the IEEE/RSJ Int. Conf. on*, 4261–4266.

Tsianos, K.I., Sucan, I.A., and Kavraki, L.E. (2007). Sampling-based robot motion planning: Towards realistic applications. *Computer Science Review*, 1(1), 2–11.

Zickler, S. and Veloso, M. (2009). Efficient physics-based planning: sampling search via non-deterministic tactics and skills. In *Proc. of The 8th Int. Conf. on Autonomous Agents and Multiagent Systems-Volume 1*, 27–33.