Linear Temporal Logic-based Mission Planning

Anil Kumar and Rahul Kala

Student Member, IEEE, Member, IEEE Robotics and Artificial Intelligence Laboratory, Indian Institute of Information Technology, Allahabd, India

***Abstract* — In this paper, we describe the Linear Temporal Logic-based reactive motion planning. We address the problem of motion planning for mobile robots, wherein the goal specification of planning is given in complex environments. The desired task specification may consist of complex behaviors of the robot, including specifications for environment constraints, need of task optimality, obstacle avoidance, rescue specifications, surveillance specifications, safety specifications, etc. We use Linear Temporal Logic to give a representation for such complex task specification and constraints. The specifications are used by a verification engine to judge the feasibility and suitability of plans. The planner gives a motion strategy as output. Finally a controller is used to generate the desired trajectory to achieve such a goal. The approach is tested using simulations on the LTLMoP mission planning tool, operating over the Robot Operating System. Simulation results generated using high level planners and low level controllers work simultaneously for mission planning and controlling the physical behavior of the robot.**

***Keywords* — Temporal Logic, Linear Temporal Logic, Mission Planning, Robot Operating System, Robot Motion Planning.**

I. Introduction

The problem of Robot Motion Planning is to enable a robot to navigate and make its way out in a complicated obstacle-prone environment. A classical motion planning problem using temporal logic [1], is described simply as "Go from A(source) to B(goal) by avoiding obstacles", where A and B may be some regions of interest in the robotic world. Since last decades robots are increasingly becoming more autonomous and are able to work in the absence of humans. Therefore simple goal specifications also become more complex and we need to plan the motion beyond a single goal problem. One of the most common dreams is that a robot should work as a *helper robot*, which obeys human instructions and never violates the robotics laws.

One must be able to robustly satisfy commands such as "Surveillance of all laboratories in any order for safety purposes", "Pick and deliver food items from cafeteria to the Robotics Laboratory and all other laboratories", "If a robot is in the robotics laboratory and any disaster occurs, then inform all research students in the laboratory and hostel". We can not always assume the navigation environment of the mobile robot to be static. It may change with the environmental conditions and can be sensed by the robot's sensors. Recently there has been increasing interest to have complicated task specifications in partially known and unstructured environments. Some common basic examples include: reach on some specific goal and convergence to that region ("reach A eventually and stay there forever"), visit a set of goals sequentially ("reach A then C then B until D is visited"), surveillance ("reach A then B and repeat the same for numerous occasions"), condition ("never go to A until regions B or C are visited"), liveliness ("move to all regions continuously"). The basic notions can be combined in numerous ways to make more complicated specifications possible. This gives a language, very similar to natural language, to the owner to control the robot. It is evident from current research that in the future robots will be able to converse with the humans and take complex specifications such as: Go to the laboratory to get the equipment from the assistant, while on the way also see how are the students doing. At some time drop a letter to Prof. X and if someone gives a letter on the way, also get that too.

Several search algorithms have been designed with complex specifications. In this paper we use Temporal Logic to represent the complex goal specifications [2] [3]. Temporal logic is a tool which can model such relations between different entities, in our case multiple regions or goal points. Temporal Logic is an extension to the conventional Logic based systems, with the difference that it can represent temporal relations which depend upon time. It is a very expressive way to define complex goal specifications with constraints, while using formal notions to enable verification checks.

The paper solves the problem of mission planning where the mission is specified by a user in terms of a LTL formula. The map comprises of regions of interest and obstacles. The map is parsed with a low level planner to get all the transitions between different regions, while satisfying any task constraints. The information is stored as an automaton. High level planning is applied on the automaton to get a strategy acceptable by the automaton, which is implemented using low level controllers.

Many mission specifications may require inputs not available or visible to the robot. As an example, consider the mission which asks a robot to go immediate to a particular place when asked for by a user, besides other specifications. The robot does not know when it would be asked to approach the place by the user and hence it can't plan in advance. Robots compute a strategy for best performance against the most adverse sensor readings, and use this strategy for navigation. The trajectory is derived from the path based on the sensor readings. It is assumed that the map with all specifications of regions of interest and obstacles is known to the robot, although obstacles may change with time, which is handled by the low level controller.

The paper is organized as follows. *Section 2* discuses and presents some related works in the literature. *Section 3* presents the overall overview of the work. *Section 4* subsequently presents a primer of LTL, which is the technique used for mission specification and planning. The specification is represented as a Bchi Automata described in *Section 5*. *Section 6* describes the simulation framework and gives the simulation results. Finally, *Section 7* gives the concluding remarks.

II. Related Works

We are always motivated by the previous research involving robotics and other related fields. Robot motion planning is one of the most challenging domains for study and research. This section very briefly describes some of the related works. In a framework presented by Gazit et al. , the authors solved the reactive mission motion planning

DOI: 10.9781/ijimai.2016.375

problem using a hybrid controller to control the behaviors of the robot based on the environmental conditions such as rescue, coverage and obstacle avoidance. In [1], firstly the authors created a feasible discrete controller for all LTL specifications, and then captured the reactive behaviors of a single or a group of robots which is computationally feasible in complex environment. Kala et al. [27] solved the problem of robot path planning in dynamic environments using a Hierarchical Evolutionary technique, wherein the robot had to find the optimal path from the entire given map. Several moving obstacles were used on the map during the motion of the robot. Hierarchical Evolutionary algorithm optimized the extra step cost in a variety of scenarios. Kala et al. [25] solved the navigating of multiple mobile robot using cell decomposition technique and reactive planning with fuzzy logic whereas indirect communication between mobile robot is achieve through simple fuzzy-logic.

A mathematical discrete-optimization based trajectory generation using Linear Temporal Logic specification was solved by Gazit et al.[1] where the author presented an optimal control for a non-linear system using LTL specification and generated a trajectory to control. Gazit et al. [2] provided an environmental model for automatically verifiable controllers that satisfy the high level task specification for the mobile robot based on the sensors/actuators information which described the local behaviors of the robot. However for computational approach discrete controllers must satisfy General Reactivity GR(1) formulas [18]. If LTL specification is feasible then only the mobile robot will be able to complete the desired task. Svorenova et al.[9] formulated a mission planning problem for the mobile robot in partitioned environment for task specifications like surveillance mission, obstacle avoidance etc. An automaton based approach was used by the authors for model checking, for high level task accomplishment by the robot and also for collecting the rewards form the visited regions.

Sampling based motion planning using temporal goals is presented by Bhatia et al. [3], where a multi-layered synergistic approach was used for all temporal goals. In this multi-layered synergistic framework, the high level discrete structure was involved for exploring the information and suggested the high level goal plans. The authors also used geometry based multi-layered approach and continuous time-step for low level sampling of the task. In another approach Bhatia et al.[4] extended their work by using nonlinear hybrid dynamics for the high level temporal goals. Moreover a lazy-search approach was used for the high level planning thus resulting in computational speedups by 10 times for the second order non-linear hybrid controller compared with the previous method. Moreover sampling based Tree-Search motion planning with discrete abstractions for specifications using temporal logic proposed by McMahon et al. [8]. In this framework the authors suggested very low cost motion trajectories that satisfy Co-Safe LTL. A physics based engine was managed which was responsible for the robots' accurate motions with their own physical dynamics. There has been additional work using multi-robot systems. Saha et al.[5] proposed a compositional framework for motion planning based on Satisfiability Modulo Theories (SMT), which are a combination of linear constrains solvable within given motion primitives and generate an optimal trajectory. Here the behaviors of the multiple robots was specified by a set of temporal logic formulas. A controller was used to trace all trajectories for each robot in the experimental workspace. Thus an optimal cost was maintained for each trajectory based on the motion primitives for that robot and synthesize-able for the given configurations.

Although all the above discussed works are mainly on completely known environments, Guo et al.[6] solved the mission planning in partially known workspaces where the task specification was given by a Linear Temporal Logic formula. In this generic model the mission specification was done by LTL based on the previous knowledge of the system environment. Further as more information about the environment model was available, the plans for the robots were revised. This type of approach is good for an environment where the robot operates in partially known workspaces. Another approach for motion planning in unknown environment was proposed by Ayala et al. [7]. The authors solved the LTL specification for an unknown environment wherein an accurate assumption of the sensors and actuators data was made, and the environment was partitioned into finite square cells. The authors used a formal method for verification and a grid based exploration method. During the authors experiments, the initiation of the algorithm started only when the robot has enough information for processing the given task specifications.

An automated LTL based task complication for a multi-robot system was proposed by Loizou et al.[10]. In this experimental framework the author suggested an input-output module used for task specification given by LTL by considering the environmental condition such as safety, liveliness and obstacle avoidance. The authors suggested Multi-Robot Navigation Function controllers which were generated by the linear temporal logic task specifications.

Several software model based frameworks are proposed to validate the task specification using model based software toolkits which synthesize the task specification generally using GR(1) and *Büchi* Autometa. An automatically synthesizing DD (Digital Design) from LTL specification was proposed by Piterman et al.[26] wherein GR(1) was used to synthesis the specification using game planning techniques, resulting in always winning strategies and plans. Thus the number of individual states may increase during the process of the algorithm, and formal specifications of the task may lead to complexity on experimental environments. Finucane et al.[11] solved the mission planning for mobile robots using the LTLMoP toolkit. Here the mission planning specification was given through temporal logic in English structured language which was synthesized by GR(1) and the mission was further validated during the motion of the robot. Sensors and actuators were used to collect real-time information from the environment. However another approach for mission planning was proposed by Jing et al.[12] where LTLMoP software was used on the given map to analyze the users specification as either valid or invalid. Moreover for demonstration of the task specification the authors used simulations and a physical robot wherein the path planning problem was solved by the Open Motion Planning Library.

Action verification during the motion of the robot, having correct-by-constructions controllers for temporal logic based synthesis, was proposed by Raman et al.[13]. Here the physical robot was controlled and invoked by low-level behaviors. If the behaviors of the low-level were unsafe or in a different time length, then a hybrid controller was used to determine whether the behaviors are safe or unsafe. All the user-friendly specifications execute at a time, one for slow and another for fast controllers and finally both are synchronized. However several specifications may also not be synthesizeable by GR(1) due to incorrect specifications and thus Raman et al.[14] presented an autonomous behavior based controller which compared a nontrivial set of actions and decided whether the given task is feasible and acceptable by a GR(1) parser or not. These kind of algorithms are more feasible for complex behaviors where the given task may not be free from deadlock/may cause failure during the implementation of the task.

Model checking and verification of any LTL specification[26] is a commonly used operation for the specific hardware and software configurations. Using TuLiP toolkit, Wolff et al. [28] interfaced several nonlinear optimal control for LTL specifications based on game solving models. A finite-state automaton was generated by the controller that evaluated the specifications. Only synthesizable tasks in the experimental environment were processed and therefore resources were optimally used. The beautify of TuLiP model checking was that it

converted the high level specification into low level smaller problems. Thus the computational complexity of task solving becomes more efficient.

Kunze et al. [15] proposed a framework wherein every-day object manipulation problems were generated through a LTL formula. A mobile robot (PR2) was used to demonstrate the feasible of solutions by using the LTL formula. All task specifications were validate in logic programming language PROLOG. The robot behaviors and action plans depended on the results generated by PROLOG using a backtracking mechanism. All the test scenarios during the experiments were divided into three parts namely grasping, pouring pancake and truing pancake. These all experiments were done with a physical robot PR2 also in ROS environments.

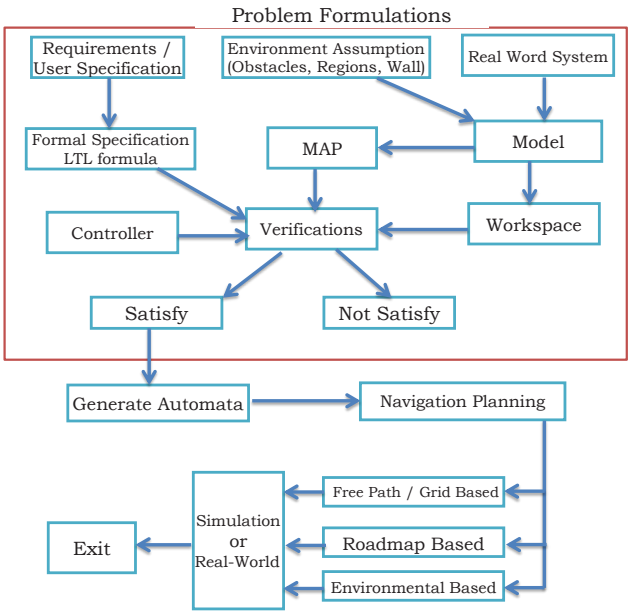

Fig. 1: Graphical Representation of Proposed Methodology

Numerous approaches have been developed to translate high-level, heterogeneous tasks to low-level continuous or discrete controllers, optimally and in a computationally efficient manner [16] [17]. Moreover Fusion of probabilistic A* and fuzzy interface system for mobile robot path planning is achieved by Kala et al.[19]. Tiwari et al.[20] present the also several algorithmic approaches for mobile robot path planning in many experimental environmental assumptions.

III. Methodology

In this section we discuses about the general outline of the proposed work of motion planning which is also represented in Fig 1. The overall approach for solving the motion planning consists of user specifications, system modeling, environment knowledge, navigation planning etc. The system methodology in our problem represents the layout of the environmental, how we model any problem related to mission planning using temporal specifications and implementation of the motion strategy. The constituents of our methodology are given below:

- **Requirements/ User specifications :** All the robot mision related query of the user is specified in this section for all approaches used in this work. The user specifications must be written in either LTL specification format or translatable into LTL symbolic specifications. However it may possible that the requirements of the user for the robot mission may be ambiguous or invalid. Thus we need to perform verification of the given problem on some LTL verification engine.
- **Environment Assumptions:** The robotic environment may consist of obstacles (i.e. non-navigation areas for the mobile robot, walls, etc.) and regions of interests (i.e. goal points, e.g. rooms, laboratories, areas, etc.). This information is the most important while preparing the robotic navigation map for the mobile robots.
- **Real Word System Modeling:** While representing the real-world system for our navigation map in workspace configurations we need efficient techniques for representing the complicated world system. Thus sometimes we need to redesign the real-world environmental information for robotic map. The path from the source to goals is fully dependent how the real-world system is represented. Thus the real-world system needs to eventually translate into system state model representation so that we can apply LTL state verification techniques for the given user specifications. The system design of any real-world scenario can be represented with $\mathcal{M}$. If the temporal logic formula ϕ holds the system proposition S_j, the system can be represented as: $(\mathcal{M}, S_j) \models \phi$.
- **Map and Workspace:** The entire real-world system of experimental scenario is represented using a map. Map of the real-world environments may be 2D or 2.5D as in our experiments. The map conversion from real-world to workspace environment can be done through camera calibration, distance information to obstacles and regions, etc. Moreover the mapping is also required for real-world map size to workspace size reductions.
- **Verification and Satisfying Conditions:** The users requirement are in the form of LTL specifications which need to be verified before initially starting the motion execution for a given problem specification. The specifications are either satisfied by the LTL verification method or the goals are not satisfied by the LTL verification method (i.e. Go to A to B until region C does not occur is interpreted as valid; whereas Go to A and don't go to A is interpreted as an invalid specification). However for verification of LTL formula we generate an automaton based on *Büchi* Automata and GR(1) parsing method for counter-strategy generations. A strategy for navigation is generated that satisfies the LTL formula.
- **Navigation Planning:** The navigation planning generates only valid LTL specifications and synthesis of the LTL specifications. The navigation planning approach has three different methods `Grid Based`, `RoadMap Based` and `Environment Based`. In Gird based approach the map is represented in a Grid format; RoadMap Based approaches decompose the environment into several path segments or produce a roadmap so that the computation time decreases and it improve the efficiency for reaching the goals. In Environmental Based approaches a local path planner is used for the roadmap decomposition and an external sensor is used for the environmental conditions while the robot is in motion.
- **Simulations and Real-world Representations:** The experiments are done with navigation planning techniques which need a representation of the map with either simulated or a physical world representation. In all our experiments we use a simulated environment as the real-world scenario for problem solving. The Grid Based approach was used with a point sized robot in aprior work [27]. The Roadmap Based approaches are presented in this work. Moreover another efficient approach with promising result is presented by Kala et al.[24] for multirobot robot motion planning using hybrid MNHS and Genetic Algorithm(GA).

In all our assumptions the system is modeled with some environmental assumptions which undergoes verification related to goal specifications. In our model checking method the LTL formula

follows the goal-based automata approach. In our experiments we used an indoor robot navigation environment, where the Lab Regions and office areas were represented as goal regions.

IV. Linear Temporal Logic

The LTL is extended from the propositional logic with some operators which are bound with the time interval factor. In real-world scenarios, the goal is not always constant and may change with time interval due to the external interference. Similarly the multiple goals in the problem of mission planning may have temporal relations among them. Thus, single/multi-goal specifications for any mobile robot can be expressed with LTL easily. Recently temporal logic is widely accepted and is gaining popularity for high level mission planning due to its expressive power, flexibility and mathematical framework. Formal verification model based systems are widely accepted in real-time environments to verify systems such as Air-Traffic Control, Autopilot Drive Mode, CPU design etc. We perform an examination of our problem definition system based on the LTL problem specifications and ensure that the given specification/task is reachable to the desired goal or the robot will not be able to reach on the given goal positions as per the current specifications. We also formulate a strategy of navigation which results in attaining the goal condition as specified in the LTL.

In our experimental work LTL is used to describe the complex mission specification for the robot from one region to another region. Here regions may be interpreted as any region of interest on the map like A, B, labs, offices, etc. An LTL formula ϕ must satisfy all specifications over the input alphabet, $\omega \rightarrow 2^{\texttt{Prop}}$ where `Prop` (Atomic Propositions) and satisfy the models at time instant i used in the LTL formula. Linear Temporal Logic can formally be described by some traditional logic operators like *conjunction* ($\wedge$), *disjunction* ($\vee$), *negation* ($\neg$), *implication* ($\rightarrow$) and *equivalence* ($\Longleftrightarrow$) whereas additional temporal operators are used including *eventually* ($\Diamond$), *next* ($\bigcirc$), *always*($\Box$) and *until* ($\mathcal{U}$). The LTL has some properties with the environment assumptions. These include:

- **Safety**: It describes those conditions which must be always satisfied (e.g. always avoid obstacles). The LTL formula *negation* ($\neg$) is used to describe the conditions which must never hold. A natural safety condition is to never collide with any obstacle, that is: $\neg(O_1 \vee O_2 \vee O_3 \vee ... \vee O_n)\mathcal{U}\mathcal{R}$, means eventually reach region $\mathcal{R}$ by avoiding all obstacles $O_i, i = 1, 2, ...n$.
- **Liveness**: This specifies the goals which must be satisfied eventually by some actions in the future (e.g. "Eventually go to the region B and visit infinitely often").
- **Sequencing**: Describes the sequence of goals to be followed in any order. The LTL formula specifies that we must visit all regions (i.e. A,B,C,D,E) as per the requirement: $\Diamond$(A $\wedge$ $\Diamond$ (B $\wedge$ $\Diamond$ (C $\vee$ $\Diamond$D))).
- **Reachability**: An individual state s_i is reachable from an initial present state if at least one path exists which takes the system from the initial state to the state s_i.

The aim is to use the realtime map to get a trajectory $\tau(t)$ which satisfies the LTL specification, based on the sensor percepts received in realtime. The initial position of the robot is assumed to be known or we can set during the problem execution, which must satisfy the LTL specification. All LTL formulas for a real-time environment are specified with the initial state $\tau(0)$ and with regions of the environment $\chi = \{\pi_1, \pi_2, \ldots, \pi_N\}$. The LTL formulas are given by the grammar.

$$\phi \quad ::= True|False|\phi|\neg\phi|\phi \vee \phi|\phi \wedge \phi|\bigcirc \phi|\phi\mathcal{U}\phi \tag{1}$$

The boolean constants (`True`, `False`) can be defined as `True` = $\phi \vee \neg\phi$ and `False` = $\neg$`True`. All the temporal and non-temporal operators are not represented in the grammar but they can be derived with these basic operators. Using the basic temporal operator, we can define additional temporal operators such as "Eventually" $\Diamond\phi = \texttt{True}\mathcal{U}\phi_1$, safety or "Always" $\Box\phi = \neg\Diamond\neg\phi$ and "Next" operator $\bigcirc\phi$. The semantics of any LTL formula can be recursively defined as:

- $\tau(t) \models_c \neg\phi$ if $\tau[t] \not\models_c \phi$
- $\tau(t) \models_c \phi_1 \vee \phi_2$ if $\tau[t] \models_c \phi_1 \vee \tau[t] \models_c \phi_2$
- $\tau(t) \models_c \phi_1 \wedge \phi_2$ if $\tau[t] \models_c \phi_1 \wedge \tau[t] \models_c \phi_2$
- $\tau(t) \models_c \bigcirc\phi$ if $\tau[t+1] \models_c \phi$
- $\tau(t) \models_c \phi_1\ \mathcal{U}\ \phi_2$ if $\exists t'' \geq t : \tau(t') \models_c \phi_1 \forall t \leq t' \leq t'' \wedge \tau(t'') \models_c \phi_2$
- $\tau(t) \models_c \Diamond\phi$ if $\exists t' \geq 0$, $\phi^{t'} \models \phi$.

`weak-until:` The operator is defined as, ϕ_1 must be true until ϕ_2 become true and thereafter there is no bound for ϕ_2, it may either be true or false. *`Until`* and *`weak-until`* $\mathcal{W}$ are dual in nature, expressed as:

$$\neg(\phi_1\mathcal{U}\phi_2) \equiv (\phi_1 \wedge \neg\phi_2)\mathcal{W}(\neg\phi_1 \wedge \neg\phi_2)$$
$$\neg(\phi_1\mathcal{W}\phi_2) \equiv (\phi_1 \wedge \neg\phi_2)\mathcal{U}(\neg\phi_1 \wedge \neg\phi_2)$$

`Release:` The release operator $\phi_1\mathcal{R}\phi_2$, is defined as, ϕ_2 must be true from starting where ϕ_1 becomes true for first time. Somehow if ϕ_1 never becomes true, then ϕ_2 needs to be true for all future times or forever. The *`release`* $\mathcal{R}$ operator can be expressed as $\phi_1\mathcal{R}\phi_2 \equiv \neg(\neg\phi_1\mathcal{U}\neg\phi_2)$. The *`until`* $\mathcal{U}$ and *`release`* $\mathcal{R}$ are also dual and can be expressed as:

$$\phi_1\mathcal{U}\phi_2 \equiv \neg\phi_1\mathcal{R}\neg\phi_2$$
$$\phi_1\mathcal{R}\phi_2 \equiv \neg(\neg\phi_1\mathcal{U}\neg\phi_2)$$

The expansion law is also satisfies by the *`release`* $\mathcal{R}$ operator that is: $\phi_1\mathcal{R}\phi_2 \equiv \phi_2 \wedge (\phi_1 \vee \bigcirc(\phi_1\mathcal{R}\phi_2))$.

The pictorially representation of LTL operators is shown in Fig 2. $\mathbf{A}$ and $\mathbf{B}$ are two propositions used to represent the region information, ϕ_1 and ϕ_2 are used for LTL problem formulations. The "Next" operator $\bigcirc\mathbf{A}$ holds the next state. Similarly $\Diamond\mathbf{A}$ represent s that will eventually hold somewhere on path a subsequent path. The $\Box\mathbf{A}$ means that the formula needs to hold all future paths from the state where it belongs. $\mathbf{A}\ \mathcal{U}\ \mathbf{B}$ means $\mathbf{A}$ has to hold on all the future states until $\mathbf{B}$ does not occur. Finally $\mathbf{A}\ \mathcal{R}\ \mathbf{B}$ presents the release operator. Based on task specification we need to formulate the LTL specification for testing goal conditions, wherein a combination of the operators are used.

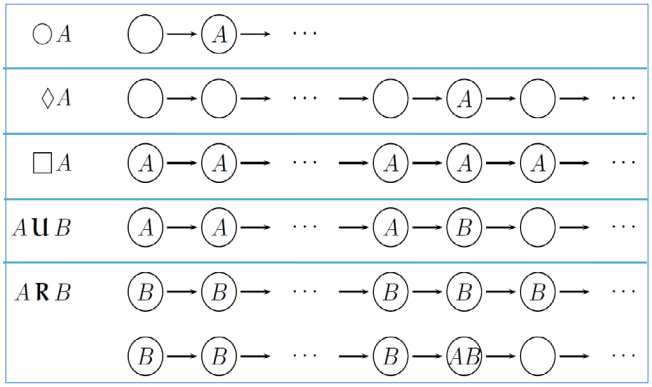

Fig. 2: Graphical Representation of LTL Operators

Syntactically the LTL formula ϕ is a Deterministic Finite Automaton (DFA) which can be easily contracted and always satisfies the finite traces[23]. The DFA for any given tuple can be defined as: $\mathcal{T}_\phi = (\mathcal{S}, \omega, \delta, S_0, S_F)$.

- $\mathcal{S}$ is a collection of all states
- $\omega \rightarrow 2^{\text{prop}}$ represents all input alphabets
- δ is the transition function, $\mathcal{S} \times \omega \rightarrow \mathcal{S}$
- $S_0 \in \mathcal{S}$ is the initial state
- S_F is a set of accepting states or the terminal state for the defined model

Run of a tuple $\mathcal{T}_\phi$ is sequence of states $s_0, s_1, \ldots, s_n$ where $s_0 = S_0$ and $\omega_i \in \mathcal{S}$ for all $i = 1, 2, \ldots, n$, moreover the tuple will accept ω only if $\omega_n \in S_F$.

The fairness constrains in LTL may be expressed with two different assumptions, either `Action-based Fairness` or `State-based Fairness`. The conclusion of using Fairness in Transition System (TS) is that we need to determine if the entire TS for any LTL formula only needs to verify the LTL formula ϕ for fairness execution of LTL formula over TS. Thus assume Φ_1 is defined as "Something is Enabled" and Φ_2 represents "Something is Taken" then the Fairness constraints can further be expressed in following ways:

`Unconditional Fairness:` The specification does not express any conditions or circumstances for something happening on some period of time. Thus the specification becomes much easier to implement.

$$ufair = \square\diamondsuit\Phi_1$$

`Strong Fairness:` Assume that the LTL specification consists of an activity *infinitely often*. E.g. the task condition "Visit IT_LAB and stay there infinitely often. The condition may become false after some time. Although sometimes our goal reversibility is not fully satisfied for all future times then `Strong Fairness` is more general for invoking LTL specifications.

$$sfair = \square\diamondsuit\Phi_1 \rightarrow \square\diamondsuit\Phi_2$$

`Weak Fairness:` The `Weak Fairness` may be presented by those LTL specifications where an activity/task is always *continuously enabled*. If the task is continuously enabled then it does not generally allow the temporary disabling.

$$wfair = \diamondsuit\square\Phi_1 \rightarrow \square\diamondsuit\Phi_2$$

Moreover Fair paths and Traces assumptions can be expressed for any set of state/actions. Let the fairness be given by $\mathcal{F} = (\mathcal{F}_u, \mathcal{F}_s, \mathcal{F}_w)$, then Fair Path and Fair Traces can be expressed as:

$$FPath(s) = \{\tau \in Paths(s) | \tau \models fair\},$$
$$FTraces(s) = \{trace(\tau) | \tau \in FPath(s)\}$$

Therefore Path from initial state $s_0 \rightarrow s_1 \rightarrow \ldots$ $is\ \mathcal{F} - fair$ only iff there exists an $\mathcal{F}$ execution $s_0 \overset{a1}{\rightarrow} s_1 \overset{a2}{\rightarrow} \ldots.$

V. Büchi Automata For LTL Specifications

Büchi automaton extends the finite automaton for infinite input word sequences. It is a variant of $\omega-$ automaton. The *Büchi* automaton accepts infinite words sequences as input for a given model and visits (at least) states from the given state to the final states infinitely often. automata is widely accepted for model checking in LTL for automata-based verification.

Formally we can define a deterministic *Büchi* automata as the quintuple $\mathcal{A} = (S, \Sigma, \delta, s_0, F)$ where the each component is defined as:

- S : is a finite set of states of $\mathcal{A}$
- Σ : is a finite set of alphabet of $\mathcal{A}$.
- δ : is $S \times \Sigma \rightarrow S$ the transition function of the tuple $\mathcal{A}$
- s_0 : is an initial state where $s_0 \in S$ of the tuple $\mathcal{A}$.
- **F** : $F \subseteq S$ is called as the acceptance conditions/final states and at least one state should occur from the set of final state $\{F_1, F_2, \ldots, F_n\}$.

Consider a run of *Büchi* automata over a finite alphabet Σ. The set of infinite words $w = w_0, w_1, \ldots, w_n \in \Sigma$ and the sequence of states are $s_0, s_1 \ldots, \in S$ such that $\forall\ \mathcal{A} \geq 0,\ \delta(s_i, w_i) = s_{i+1}$. The set of infinite words(w) for which we visit at least one state in F are accepted words and can be denoted by $\mathcal{L}_w(\mathcal{A})$. A *Generalized Büchi Automaton* (GBA) is used where the number of accepting state increase $F = F_1, F_2, \ldots, F_k$, where k is the number of final states. Therefore GBA will accept any set of words iff, for all $1 \leq \text{i} \leq \text{k},\ S \cap F_i \neq \phi$.

We can also express the extended transition relations δ^w: $S \times \Sigma^w \times S$. This relation shows that it will take as input the set of stings rather then single characters. Let us assume that an empty string is represented by ζ, then δ^w can be defined recursively as $\forall s \in S,\ w \in \Sigma^w,\ \sigma \in \Sigma$ such as:

$$\delta^w(s, \zeta) = s$$
$$\delta^w(s, w\sigma) = \delta(\delta^w(s, w), \sigma).$$

Intuitively, let us assume that $\mathcal{A}_M$ and $\mathcal{A}_N$ are two different states running in parallel and synchronously to each other, then σ is an event which is associated from the state s_0 to s_0' from model $\mathcal{A}_M$ to $\mathcal{A}_N$ respectively at time $t = 0$ to $t = 1$ times. Thus a computation of LTL formula ϕ is a set of infinite words that can be represented by the alphabet $\Sigma = 2^{PROP}$, which need to accepted by the *Büchi* automata $\mathcal{A}_w$ for all sets of input sequence ϕ.

A. LTL to GNBA States:

Loosely speaking in formal verification methods, FSM based models need to express an equivalent *Büchi* automata for LTL formula. Generally speaking LTL formula is converted by $\omega - language$ [22] for equivalent GNBA specifications . However to convert any LTL formula into corresponding *Büchi* automata we need to find the *closure* of the given system specifications. Thus a closure of any formula ϕ can be found as $closure(\phi) = \{sub - formula\}$ of ϕ and $\{Negationofsub - formula\}$ of ϕ. However let us assume that a LTL formula $\phi = a \wedge b$ then its subset is $\{a,\ b,\ a \wedge b, \zeta\}$ $\cup\ \{\neg a,\ \neg b,\ \neg(a \wedge b), \neg\zeta\}$ and also assume that the subset of closure (ϕ) is represented by B_i, then there are some elementary set of formulas for determining the property of the GNBA represented as:

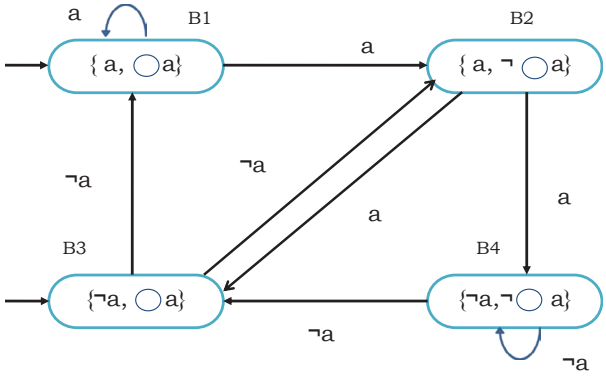

Fig. 3: LTL formula($\bigcirc\ a$) in GNBA

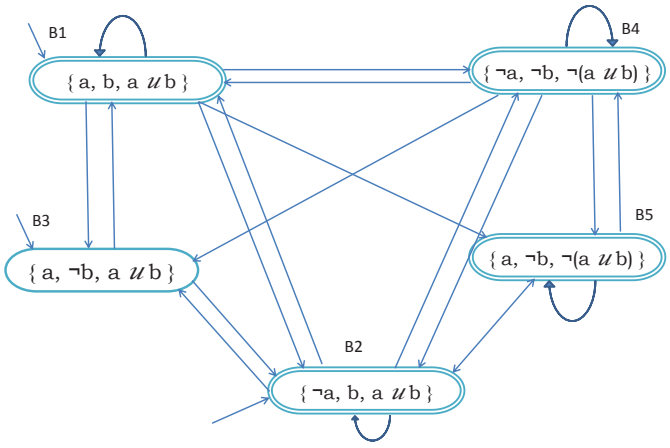

Fig. 4: LTL formula($a \mathcal{U} b$) in GNBA

1. B is logically equivalent/consistent if for all $\phi_1 \wedge \phi_2,\ \zeta \in cl(\phi)$
   - if $\phi_1 \wedge \phi_2 \in B \iff \phi_1 \in B$ and $\phi_2 \in B$
   - $\zeta \in B \Rightarrow \neg\zeta \notin B$
   - $true \in cl(\phi) \Rightarrow true \in B$
2. B is locally equivalent/consistent if for all $\phi_1 \mathcal{U} \phi_2 \in closure(\phi)$
   - $\phi_2 \in B \Rightarrow \phi_1 \mathcal{U} \phi_2 \in B$
   - $\phi_1 \mathcal{U} \phi_2 \in B$ and $\phi_2 \notin B \Rightarrow \phi_1 \in B$
3. B is called maximal, if for all $\zeta \in closure(\phi)$, $\zeta \notin B \Rightarrow \neg\zeta \in B$

However in Fig 3 we present an automaton for the LTL formula $\bigcirc\ a$ to GNBA. The terminal a may be interpreted as any region on the given map or goal specifications as per requirements. Thus the closure of $\bigcirc\ a = \{a, \neg a, \zeta\}$ and similarly in another example we present an automaton of the LTL formula $a \mathcal{U} b$ in GNBA. Moreover the formula $a \mathcal{U} b$ similarly may interpreted as "Visit IT_ LAB until visit CS_LAB". However the closure for the same formula is displayed in Fig 4 with closures of the given specifications.

B. GNBA to GR(1) Parsing :

GR(1) parsing is generally based on two-player games played in some environment. The major goal for the parsing technique is to find the winning conditions for the LTL formula over the GNBA automaton. The goal of the system depends on the environmental actions performed while moving from one system state to another system state. A general Game structure for GR(1) can be defined as $G = (\mathcal{R}, \mathcal{I}, \mathcal{O}, \mathcal{Q}_{env}, \mathcal{Q}_{st}, \mathcal{T}_{env}, \mathcal{T}_{st}, \omega)$ and is expressed as:

- $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$: A set of all *state variables* or a finite set over the finite domains. We also assume that all sets of states ($\mathcal{R}$) are Boolean from the environment $\mathcal{M}_{\mathcal{R}}$ for computation purposes.
- $\mathcal{I} \subseteq \mathcal{O}$ is a set of *input variables* for an environment which is controlled during the motion of the robots.
- $\mathcal{O} = \frac{\mathcal{R}}{\mathcal{I}}$ is a set of *output variables* controlled by the system $\mathcal{M}$.
- $\mathcal{Q}_{env}$ is boolean assertions for $\mathcal{I}$ and make distinct the initial states of the environment.
- $\mathcal{Q}_{st}$ is also boolean assertions for $\mathcal{O}$ and make distinct the initial states of the systems $\mathcal{M}$
- $\mathcal{T}_{env}(\mathcal{R}, \mathcal{I}')$ is the transition relation for the environment. This assertion is satisfied by any state $s \in \mathcal{M}$ to possible next input for $s_{\mathcal{I}} \in \mathcal{M}$ at the time interval t to $t+1$ by putting $\mathcal{I}$. Thus transition relation $\mathcal{T}_{env}$ is $(s, s_{\mathcal{I}}) \models \mathcal{T}_{env}$.
- $\mathcal{T}_{st}(\mathcal{R}, \mathcal{I}', \mathcal{O}')$ is the transition relation for the system. This assertion is satisfied for any relation for the output value for $s_{\mathcal{O}} \in \mathcal{M}_{\mathcal{O}}$ for all combinations of $\mathcal{R}$. Thus the transition relation $\mathcal{T}_{st}$ is identifies as $s_{\mathcal{O}} \in \mathcal{M}_{\int \sqcup}$. Therefore state s will read the input $s_{\mathcal{I}}$ if $(s, s_{\mathcal{I}}, s_{\mathcal{O}}) \models \mathcal{T}_{st}$.
- ω is the winning conditions for any given LTL formula orver the GNBA automaton.

However any initial state s must satisfy both $\mathcal{Q}_{env}$ and $\mathcal{Q}_{st}$ i.e. $s \models \mathcal{Q}_{env} \wedge \mathcal{Q}_{st}$. Any play between two states s and s' respectively is a successor of s only if $(s, s') \models \mathcal{Q}_{env} \wedge \mathcal{Q}_{st}$. Fig 5 shows a small example of GR(1).

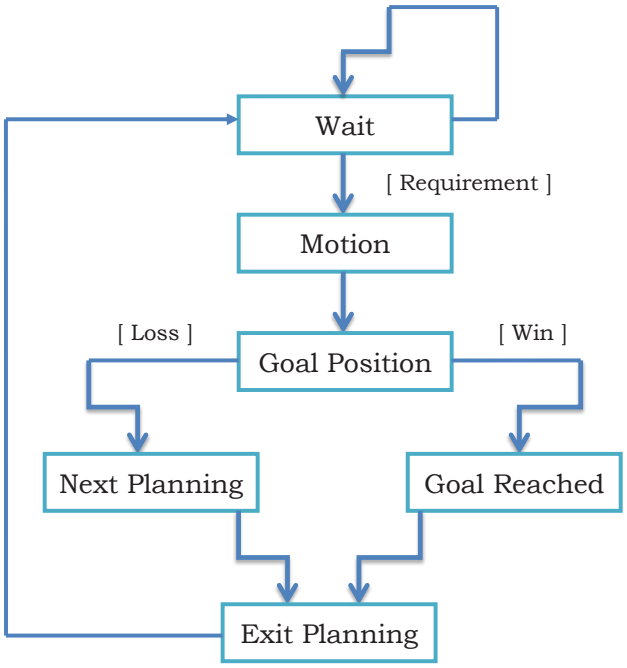

Fig. 5: GR(1) parsing example for LTL formula

VI. Simulation Results

The simulations have been done in the LTLMoP (Linear Temporal Logic MissiOn Planning) software [21]. All our experiments are performed under Ubuntu 12.04 LTS operating system environments with Intel Core i5-5200GHz processor, 4GB RAM on a desktop computer. The software allows the user to write LTL specifications as well as define the map of the environment. We setup the map based on the lab environment of IIIT Allahabad Robotics and Artificial Laboratory, although changes were made to make the environment complex and the display more appealing. Several complex labs are setup for experimental simulation purpose. Although the current setup was 2D, LTLMoP also supports 2.5D regions. All real-time information about the current experiments and motion status of the robot with all physical information can visualized in the simulation tool.

The first set of experiments are done on a 2D map of the laboratories, in which some obstacles are added. A point robot is taken that needs to visit several places on the map based on conditional requirements given by the user as the problem specifications. The map and the regions are illustrated in Fig 6. To solve the problem we setup some custom propositions corresponding to each regions shown in Table I.

First we apply roadmap decomposition on the map. The decomposed map is shown in Fig 7.

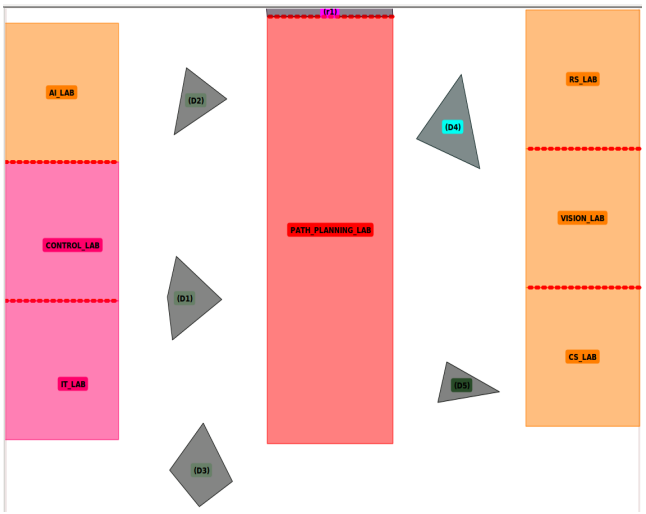

Fig. 6: Artificial map for experimental environment

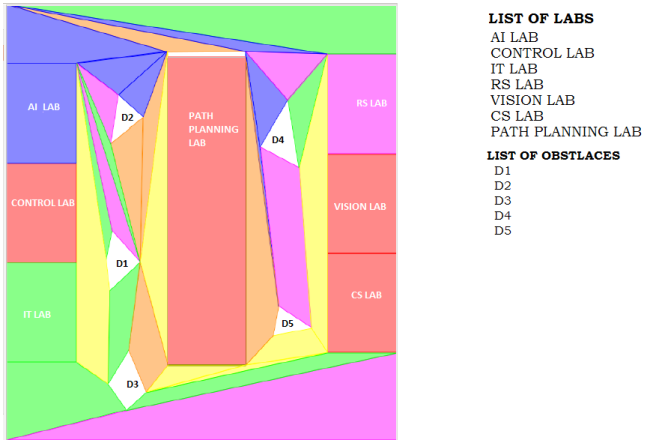

Fig. 7: Workspace Decomposition of experimental map

TABLE 1. REGION DECOMPOSITION INTO SYSTEM PROPOSITION

MAP REGIONS	CUSTOM PROPOSITIONS
IT_LAB	it_lab_visited
CS_LAB	cs_lab_visited
VISION_LAB	vision_lab_visited
RS_LAB	rs_lab_visited
PATH_PLANNING_LAB	path_planning_lab_visited
CONTROL_LAB	control_lab_visited
AI_LAB	ai_lab_visited
	x,y

In the first scenario the robot is asked to visit either `IT_LAB` or `AI_LAB` and then `RS_LAB` and `CS_LAB` until `CONTROL_LAB` is not finally visited. The LTL specification for the same problem interpreted as $\bigcirc(\bigcirc(IT\_LAB \vee AI\_LAB) \wedge \bigcirc(RS\_LAB \wedge CS\_LAB))\, \mathcal{U}\, CONTROL\_LAB$. The mission specification is given by Algorithm 1. This is a common problem in robotics, wherein the robot may be asked to talk to any student sitting in either of the two labs, and then broadcast the information to all the other labs, finally ending the journey at the home lab. The resultant trajectory displayed on the map in black color is shown on Fig 8.

Algorithm 1 Robot Waiter Example

```
ai_lab_visited is set on AI_LAB and reset on false
rs_lab_visited is set on RS_LAB and reset on false
it_lab_visited is set on IT_LAB and reset on false
cs_lab_visited is set on CS_LAB and reset on false
x is set on (it_lab_visited or ai_lab_visisted) and reset on false
y is set on (rs_lab_visited and cs_lab_visisted) and reset on false
if x then
    Go to y
    Go to x
end if
```

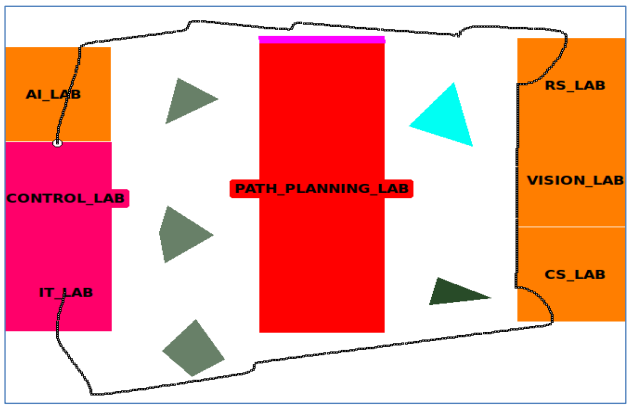

Fig. 8: Results for First Scenario

In the second scenario the robot is ask to perform some search or rescue/surveillance task in our experimental environment and it needs to visit all labs on map. In this specification we take an assumption that the robot needs to visit all the research labs in any order. The specific LTL specification is represented as: $\bigcirc(CS\_LAB \wedge VISION\_LAB \wedge RS\_LAB \wedge AI\_LAB\, CONTROL\_LAB \wedge IT\_LAB \wedge PATH\_PLANNING\_LAB)$. The resulted trajectory is shown in Fig 9. In the third scenario the robot is asked to carry some notebooks or research papers either from IT_LAB or VISION_LAB and then visit CONTROL_LAB and AI_LAB and finally visit the IT_LAB. The corresponding LTL specification interpreted as . The trajectory traced by the robot is shown in 10.

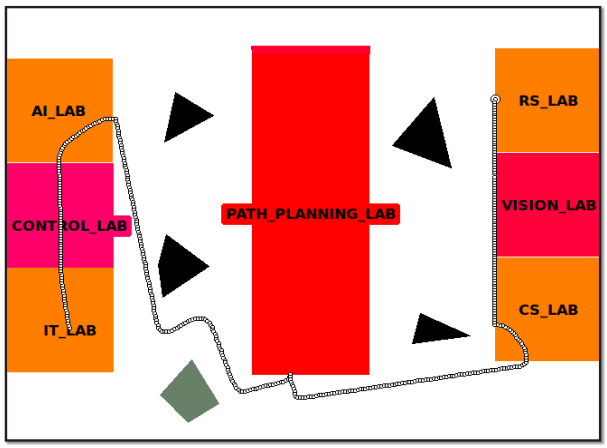

Fig. 9: Results for Second Scenario

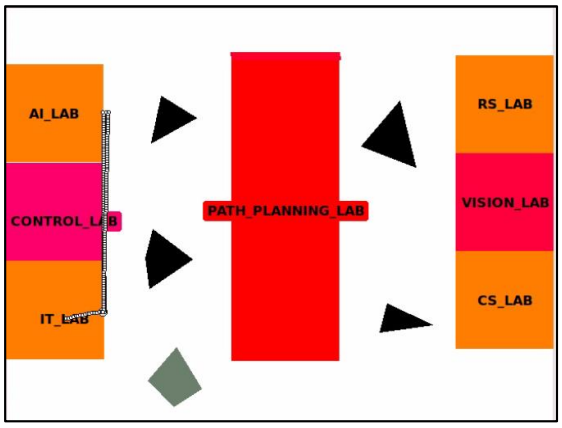

Fig. 10: Results for Third Scenario

We also setup an experimental stationary environment for a PR2 robot using LTLMoP-ROS software. The simulation result is performed under the GAZEBO-ROS GUI API. The map represents an office like environment consisting of three regions office, Lab1 and Lab2, defined as regions of interest. We also use a dummy sensor "wait", during the motion of the PR2 robot. If the dummy sensor is activated, then the robot will wait on its position infinity until the sensor is not deactivated. Fig 11 shows the map used for the experiment, while the decomposed map is shown in Fig. 12. In this scenario the robot is asked to move from initially office to Lab1 or Lab2 . The PR2 robot will only be in motion condition if the dummy sensor is disabled. The movement of the robot is shown in Figs 13, 14 and 15. Moreover as the robot visits in the office area or in a corridor a local planner will "foldArm". In Fig 16 during the motion of PR2 the robot waits as the wait sensor is enabled.

Algorithm 2 Robot Helper Example

```
robot start in office with false
initially often not wait
Visit Lab2 or Lab1
Visit Office
if you were in office then
    Do foldArms
end if
if you are sensing wait then
    Stay there
end if
```

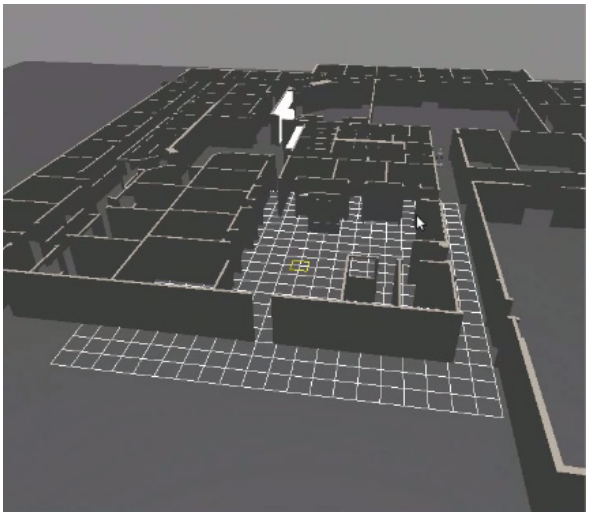

Fig. 11: Workspace of the experimental map in ROS

Fig. 12: Workspace Decomposition in ROS for PR2 Robot

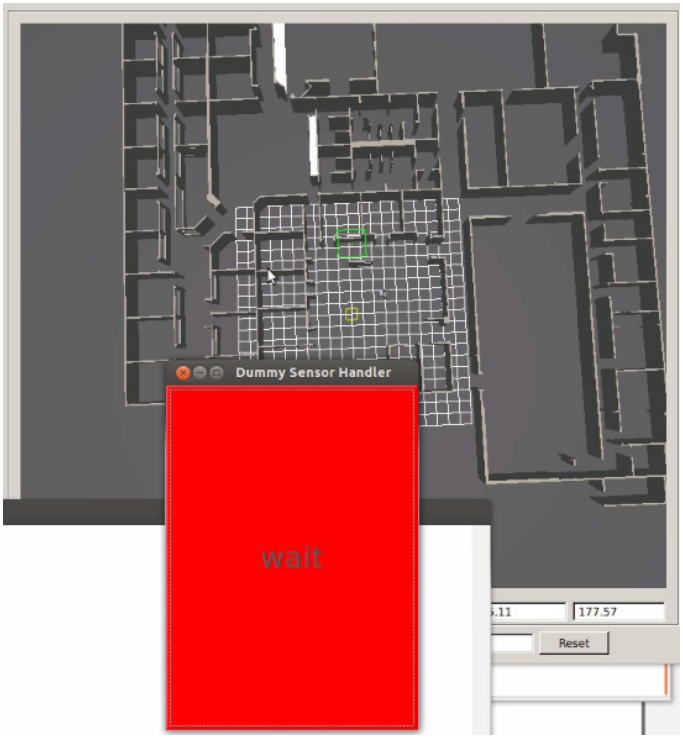

Fig. 13: PR2 robot is initially in the office area and is waiting for the sensor to be disabled

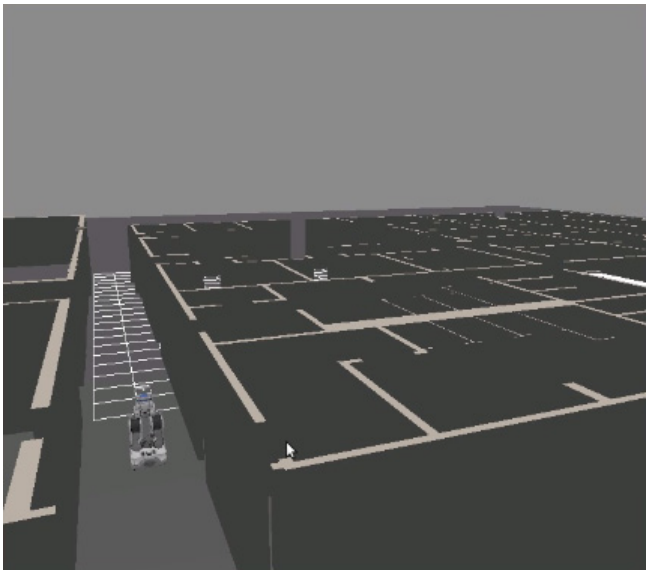

Fig. 14: PR2 robot visiting from office area to Lab1 with the Wait Sensor disabled

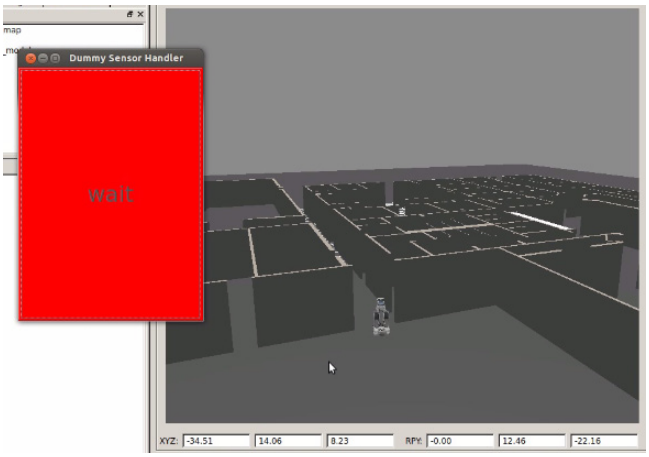

Fig. 15: PR2 robot visited Lab1 and the Wait sensor was disable

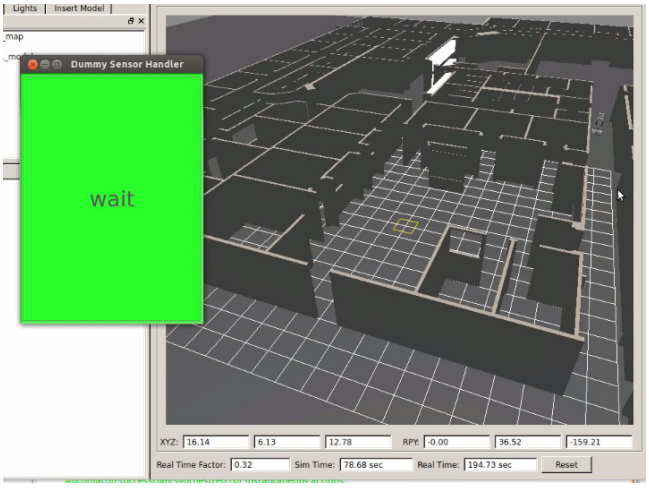

Fig. 16: PR2 robot standing on the corridor as the wait sensor was enabled

Moreover the video results of simulation can be found at:
https://www.youtube.com/watch?v=7rvbnd8MlWc,
https://www.youtube.com/watch?v=gSwOi-lJiuA,
https://www.youtube.com/watch?v=vJ2rdU2aLuE.

VII. Conclusion and Future Scope

Mission planning is a complicated problem in robot motion planning, wherein the task required to be done by the robot can be overly complicated. In this paper we used Temporal Logic for solving the problem of Mission Planning for mobile robots. The map was decomposed into regions which made the roadmap, and the approach was thus faster than the prior approach which used grid maps. The benefit of using a decomposition of the map is that the robot follows the trajectory given by the roadmap during the motion from one region to another, rather than computing the inter-region trajectory in real time. The simulations were initially done on a point robot. Finally a demonstration on ROS environment for real-time problem specifications using a PR2 robot was provided. The simulations used sensors mounted on the PR2 robot.

Currently the experiments are done with a single robot only. The work needs to be extended for multiple robots. The use of multi-resolution techniques and sampling based search techniques can significantly improve the computational time of the algorithm, while making more realistic assumptions about the environment. The algorithm also needs to be extended to handle probabilistic behavior and vague specifications. The algorithm needs to be tested on physical robots preferably with the ROS environment.

References

[1] Kress-Gazit, H.; Fainekos, G.E.; Pappas, G.J., "Temporal-Logic-Based Reactive Mission and Motion Planning," Robotics, IEEE Transactions on , vol.25, no.6, pp.1370,1381, Dec. 2009 doi: 10.1109/TRO.2009.2030225

[2] Kress-Gazit, H.; Fainekos, G.E.; Pappas, G.J., "Where's Waldo? Sensor-Based Temporal Logic Motion Planning," Robotics and Automation, 2007 IEEE International Conference on , vol., no., pp.3116,3121, 10-14 April 2007 doi: 10.1109/ROBOT.2007.363946

[3] Bhatia, A.; Kavraki, L.E.; Vardi, M.Y., "Sampling-based motion planning with temporal goals," Robotics and Automation (ICRA), 2010 IEEE International Conference on , vol., no., pp.2689,2696, 3-7 May 2010 doi: 10.1109/ROBOT.2010.5509503

[4] Bhatia, A.; Kavraki, L.E.; Vardi, M.Y., "Motion planning with hybrid dynamics and temporal goals," Decision and Control (CDC), 2010 49th IEEE Conference on , vol., no., pp.1108,1115, 15-17 Dec. 2010 doi: 10.1109/CDC.2010.5717440

[5] Saha, I.; Ramaithitima, R.; Kumar, V.; Pappas, G.J.; Seshia, S.A., "Automated composition of motion primitives for multi-robot systems from safe LTL specifications," Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on , vol., no., pp.1525,1532, 14-18 Sept. 2014 doi: 10.1109/IROS.2014.6942758

[6] Meng Guo; Johansson, K.H.; Dimarogonas, D.V., "Revising motion planning under Linear Temporal Logic specifications in partially known workspaces," Robotics and Automation (ICRA), 2013 IEEE International Conference on , vol., no., pp.5025,5032, 6-10 May 2013 doi: 10.1109/ICRA.2013.6631295

[7] Ayala, A.I.M.; Andersson, S.B.; Belta, C., "Temporal logic motion planning in unknown environments," Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on , vol., no., pp.5279,5284, 3-7 Nov. 2013 doi: 10.1109/IROS.2013.6697120

[8] McMahon, J.; Plaku, E., "Sampling-based tree search with discrete abstractions for motion planning with dynamics and temporal logic," Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on , vol., no., pp.3726,3733, 14-18 Sept. 2014 doi: 10.1109/IROS.2014.6943085

[9] Svorenova, M.; Tumova, J.; Barnat, J.; Cerna, I., "Attraction-based receding horizon path planning with temporal logic constraints," Decision and Control (CDC), 2012 IEEE 51st Annual Conference on , vol., no., pp.6749,6754, 10-13 Dec. 2012 doi: 10.1109/CDC.2012.6426041

[10] Loizou, S.G.; Kyriakopoulos, K.J., "Automated Planning of Motion Tasks for Multi-Robot Systems," Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on , vol., no., pp.78,83, 12-15 Dec. 2005 doi: 10.1109/CDC.2005.1582134

[11] Finucane, C.; Gangyuan Jing; Kress-Gazit, H., "LTLMoP: Experimenting with language, Temporal Logic and robot control," Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on , vol., no., pp.1988,1993, 18-22 Oct. 2010 doi: 10.1109/IROS.2010.5650371

[12] Gangyuan Jing; Finucane, C.; Raman, V.; Kress-Gazit, H., "Correct high-level robot control from structured English," Robotics and Automation (ICRA), 2012 IEEE International Conference on , vol., no., pp.3543,3544, 14-18 May 2012 doi: 10.1109/ICRA.2012.6225161

[13] Raman, V.; Finucane, C.; Kress-Gazit, H., "Temporal logic robot mission planning for slow and fast actions," Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on , vol., no., pp.251,256, 7-12 Oct. 2012 doi: 10.1109/IROS.2012.6385935

[14] Raman, V.; Kress-Gazit, H., "Explaining Impossible High-Level Robot Behaviors," Robotics, IEEE Transactions on , vol.29, no.1, pp.94,104, Feb. 2013 doi: 10.1109/TRO.2012.2214558

[15] Kunze, L.; Dolha, M.E.; Beetz, M., "Logic programming with simulation-based temporal projection for everyday robot object manipulation," Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on , vol., no., pp.3172,3178, 25-30 Sept. 2011 doi: 10.1109/IROS.2011.6094743

[16] Kloetzer, M.; Belta, C., "A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications," Automatic Control, IEEE Transactions on , vol.53, no.1, pp.287,297, Feb. 2008 doi: 10.1109/TAC.2007.914952

[17] Karaman, S.; Frazzoli, E., "Complex mission optimization for Multiple-UAVs using Linear Temporal Logic," American Control Conference, 2008 , vol., no., pp.2003,2009, 11-13 June 2008 doi: 10.1109/ACC.2008.4586787

[18] Alur, R.; Moarref, S.; Topcu, U., "Counter-strategy guided refinement of GR(1) temporal logic specifications," Formal Methods in Computer-Aided Design (FMCAD), 2013 , vol., no., pp.26,33, 20-23 Oct. 2013 doi: 10.1109/FMCAD.2013.6679387

[19] Kala Rahul, Anupam Shukla, and Ritu Tiwari. "Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning." Artificial Intelligence Review 33.4 (2010): 307-327.

[20] Tiwari, Ritu, ed. Intelligent Planning for Mobile Robotics: Algorithmic Approaches: Algorithmic Approaches. IGI Global, 2012.

[21] C. Finucane, G. Jing, and H. Kress-Gazit. LTLMoP website. http://ltlmop.github.com/, 2015

[22] Anping He; Jinzhao Wu; Lian Li, "An Efficient Algorithm for Transforming LTL Formula to Büchi Automaton," Intelligent Computation Technology and Automation (ICICTA), 2008 International Conference on , vol.1, no., pp.1215,1219, 20-22 Oct. 2008 doi: 10.1109/ICICTA.2008.297

[23] Kiam Tian Seow; Ming Gai; Tong Lee Lim, "A Temporal Logic Specification Interface for Automata-Theoretic Finitary Control Synthesis," Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on , vol., no., pp.565,571, 18-22 April 2005 doi: 10.1109/ROBOT.2005.1570178

[24] Kala Rahul. "Multirobot Motion Planning using Hybrid MNHS and Genetic Algorithms" Applied Artificial Intelligence 27.3 (2013): 170-198.

[25] Kala Rahul. "Navigating Multiple Mobile Robots without Direct Communication." International Journal of Intelligent Systems 29.8 (2014): 767-786.

[26] Piterman, Nir, Amir Pnueli, and Yaniv Saâ€™ar. "Synthesis of reactive (1) designs." Verification, Model Checking, and Abstract Interpretation. Springer Berlin Heidelberg, 2006.

[27] R. Kala, A. Shukla, R. Tiwari (2010) Dynamic Environment Robot Path Planning using Hierarchical Evolutionary Algorithms. Cybernetics and Systems, 41(6): 435-454.

[28] Wolff, E. M., Murray, R. M. (2013). Optimal control of nonlinear systems with temporal logic specifications. In Proc. of Int. Symposium on Robotics Research.

Dr. Rahul Kala received the B.Tech. and M.Tech. degrees in information technology from the Indian Institute of Information Technology and Management, Gwalior, India in 2010. He received his Ph.D. degree in cybernetics from the University of Reading, UK in 2013. He is currently working as an Assistant Professor in the Indian Institute of Information Technology, Allahabad, India in the Robotics and Artificial Intelligence Laboratory. He is the author of three books and over 70 papers. His recent book is on robotic planning is entitled
Intelligent Planning for Mobile Robotics: Algorithmic Approaches (IGI-Global Publishers, 2013). He is a recipient of the Commonwealth Scholarship and Fellowship Program from the UK Government; the Lord of the Code Scholarship from RedHat and the Indian Institute of Technology Bombay; and the GATE scholarship from the Ministry of Human Resource Development, Government of India.

Dr. Anil Kumar received the M.Tech degrees in information technology from Indian In- stitute of Information Technology Allahabad, India in 2015. He is author of more then four research papers in the field of Robotics and Artificial Intelli- gence. He is the recipient of the GATE scholarship from the Ministry of Human Resource Development, Government of India.