Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online
CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-5 (2016) 091–096

An LTL-Based Motion and Action Dynamic Planning Method for Autonomous Robot ⋆

Ning Xu * **Jie Li** * **Yifeng Niu** * **Lincheng Shen** *

* *College of Mechatronic Engineering and Automation, National University of Defense Technology, Changsha, China (e-mail: leonlee2009@163.com).*

Abstract: LTL-based languages provide a formal and expressive way to specify high-level complex motion and action tasks for autonomous robots. Based on LTL task specification, a method to dynamically generate motion and action sequences for a robot is proposed. Firstly, the environment knowledge, robot's motion and action capabilities are described respectively based on the partitioned regions. Secondly, a weighted finite transition system is constructed from these models to capture the robot's full functionality and the knowledge of environment. Thirdly, a dynamic planning framework is put forward considering environment update or operations failure. Finally, a fire-fighting case is simulated to demonstrate the effectiveness of the approach.

© 2016, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: LTL, Motion and action planning, Dynamic mission planning.

1. INTRODUCTION

Recently, a great amount of attention has been devoted to temporal-logic-based motion and action planning for autonomous robot to execute complex mission. Based on the abstraction of robot system (Alur et al. (2000)), missions are specified as temporal logic formula such as Linear Temporal Logic(LTL) and Computation Tree Logic(CTL)(Karaman and Frazzoli (2008); Partovi and Lin (2014)). Many of the suggested solutions utilize finite state transition systems(TS) for abstraction (Johansson et al. (2013); Tumova and Dimarogonas (2014); Ulusoy et al. (2012); Guo et al. (2013)), and the discrete plan is synthesized leveraging ideas from model checking methods (Baier and Katoen (2008)), which will be executed by continuous controllers later. Transition system can efficiently capture the behavior of the system, and reduce the complexity of controller design. Most of existing work addressed the problem in motion planning, and the transition system is constructed from environment partitioning (Fainekos et al. (2005); Johansson et al. (2013); Kloetzer and Mahulea (2015)).

For autonomous robot, the mission includes not only motion tasks but also action tasks, i.e., executing certain action under specific conditions. Therefore, there is need for taking actions into account when constructing transition system. Some relevant work can be found that integrates motion planning and action planning. In Chen et al. (2012), desired action only need to be executed in specific region, thus the action task could be represented by motion task directly. In Kress-Gazit et al. (2009), actions can be performed at any region, independent propositions are created for each action. In Guo et al. (2013), an approach is proposed to combine model-checking-based motion planning with action planning using action description languages like STRIPS. The approach gives the robot's mobility abstraction M and action map B, which are modeled by states and transitions respectively. Then, the action map is composed with each region of M, giving a complete description of robot's functionalities. It has a more practical description for actions, which gives a good reference for approach proposed in this paper. However, the actions in mission may have effects that changing the system beyond partitioned region or permanently, i.e., action done in one region may cause effect to another region, which can not be handled by above work.

In another aspect, for problems of practical interest, the environment is changeable and the robot should react to actual observations. Some existing work considers the case when the environment is partially observed. In Chen et al. (2012), the dynamic environment can be incrementally learned by automata learning method, which combined with Markov Decision Process (MDP) to find optimal plan afterwards. Jones et al. (2013) defined Distribution Temporal Logic for stochastic system with partial state information. Instead of finding off-line plan maximizing the probability of satisfaction, Johansson et al. (2013) proposed an on-line approach to plan dynamically. A preliminary plan is synthesized based on initial state and initial knowledge of environment and the plan is verified and revised with real-time observation.

In this paper, a method is proposed to dynamically generate motion and action sequences for an autonomous robot. Instead of giving the TS directly, the environment knowledge, robot's motion and action capabilities are described respectively based on the partitioned environment. Secondly, a weighted transition system is constructed from these models to capture the robot's full functionality and knowledge of the environment. Thirdly, a dynamic planning framework is put forward considering environment update or operations failure. Finally, a fire-fighting case

⋆ This work was supported by National Natural Science Foundation of China (61403410).

2405-8963 © 2016, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.07.095

is simulated to demonstrate the effectiveness of the approach.

The rest of the paper is organized as follows. The essential preliminaries are briefly introduced in section 2. The robot's motion and action planning problem are modeled in section 3, and the solution to this problem is presented in section 4. In section 5, a simulation case study is presented to illustrate the proposed approach. Conclusions and directions for future research are summarized in section 6.

2. PRELIMINARIES

2.1 Linear temporal logical and Büchi automaton

Definition 1. (Linear Temporal Logical). Over the atomic propositions Π, an Linear Temporal Logical φ can be defined inductively as follows (Baier and Katoen (2008)):

$$\varphi ::= \text{true}|\alpha|\varphi_1 \vee \varphi_2|\neg\varphi| \bigcirc \varphi|\varphi_1 \cup \varphi_2 \tag{1}$$

Where, $\alpha \in \Pi$ is an atomic proposition, $\neg$ (negation), $\vee$(disjunction) are standard boolean connectives, and $\bigcirc$ (next) and $\cup$ (until) are temporal operators. The formula, i.e., $\bigcirc\varphi$ means next clock proposition φ will be true, $\varphi_1 \cup \varphi_2$ means φ_1 will hold true until φ_2 being true.

The full power of LTL is obtained using above propositional and temporal operators. But for convenience, some other operators such as conjunction ($\wedge$), implication ($\Rightarrow$), equivalence ($\Leftrightarrow$), finally ($\Diamond$), and always($\Box$)are derived as follows:

- $\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$
- $\varphi_1 \Rightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$
- $\varphi_1 \Leftrightarrow \varphi_2 = (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$
- $\Diamond\varphi = \text{true} \cup \varphi$
- $\Box\varphi = \neg\Diamond\neg\varphi$

The LTL languages have a strong power for mission specification, i.e., an persistent surveillance mission can be specified as follows:

$$\begin{aligned}\varphi_{gather} =& (\Box\Diamond(gather)) \\ &\wedge(\Box(gather \Rightarrow \bigcirc(\neg gather \cup r_2))) \\ &\wedge(\Box(gather \Rightarrow r_1))\end{aligned}$$

Where proposition "*gather*" means gathering data and r_1, r_2 represent regions. This specification requires the robot gathering data at region one persistently and going to region two after each gathering.

Definition 2. (Büchi automaton). Büchi automaton is a tuple (Baier and Katoen (2008)):

$$A = (Q, Q_0, \Sigma, \delta, F) \tag{2}$$

Where Q is a finite states set, $Q_0 \subseteq Q$ is the set of initial states, Σ is an input alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is a non-deterministic transition relation, $F \subseteq Q$ is a set of accepting states.

Definition 3. (Run of Automaton). A run of the automaton over a given input word $\omega = \omega_0\omega_1...$ is a sequence $r = s_0s_1...$ such that $s_0 \subseteq Q_0$ and $(s_k, \omega_k, s_{k+1}) \in \delta$, for any $k \geq 0$.

The transition relation of automaton A may be nondeterministic, i,e,. same input alphabet to the same state may cause different transitions. Thus one input word

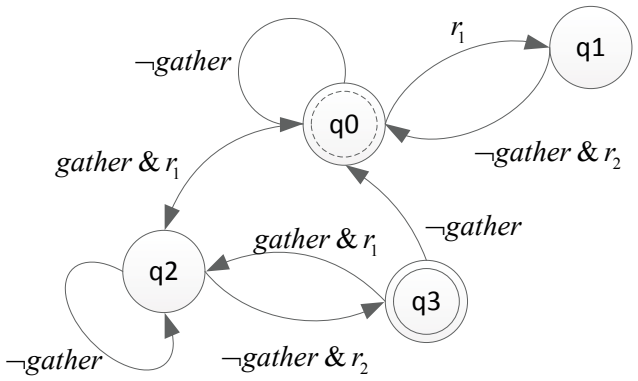

Fig. 1. Büchi automaton corresponding to φ_{gather}. "q0" is the initial state and "q3" is the accepting state. (The translating tool is "LTL2BA" (Gastin and Oddoux (2001)), some infeasible transitions are omitted such as "$r_1\&r_2$")

corresponding to several sequences (runs). An infinite word over Σ is called accepted to the Büchi automaton if one of the corresponding runs intersects with the accepting states infinitely often. This kind of runs has a common prefix-suffix structure, the prefix part R_{pre} from Q_0 to F, and the suffix part R_{suf} from F to F. The accepted run can be represented as follows:

$$\begin{aligned}R &= (R_{pre}, R_{suf}) = R_{pre}[R_{suf}]^{\omega} \\ &= q_0q_1q_2...q_{f-1}q_f(q_{f+1}...q_nq_f)^{\omega}\end{aligned}$$

For any LTL formula φ over atomic propositions Π, there exists one büchi automaton over Σ=2^{Π}, which accepts all and only words satisfy φ.The formula φ_{gather} can be translated as büchi automaton shown in Fig. 1.

2.2 Transition System

Definition 4. (Transition System). A (weighted) transition system (TS) is a tuple (Baier and Katoen (2008)):

$$T = (S, Act, \rightarrow, S_0, AP, L, W) \tag{3}$$

Where, S is the set of states, Act is the set of actions, $\rightarrow \subseteq S \times Act \times S$ is the transition relation, $S_0 \subseteq S$ is the initial states set, AP is the atomic proposition set, $L : S \rightarrow 2^{AP}$ is a labeling function giving the set of all atomic propositions satisfied in a state, W is a map assigning a positive weight to each transition. TS is called finite if S, Act, and AP are finite.

A data gather robot in partitioned environment (region R_1 and R_2) can be abstracted as Fig. 2.

Definition 5. (Product Automaton). The product automaton $A_p = T \otimes A_\varphi$ between the transition system T and Büchi automaton A_φ is defined as a tuple (Baier and Katoen (2008)):

$$A_p = (Q_p, Act, \rightarrow_p, Q_{p0}, F_p, W_p) \tag{4}$$

Where, $Q_p = \{(s,q)\,|s \in S, q \in Q\}$ is the set of states, Act is the action set of T, $\rightarrow_p$ is the transition relation, and $((s,q),(s',q')) \in \rightarrow_p$ iff $\exists act \in Act$, s.t. $(s, act, s') \in \rightarrow$ and $(q, L(s'), q') \in \delta$, $Q_{p0} = \{(s,q)\,|s \in s_0, q \in Q_0\}$ is the initial states set, $F_p = \{(s,q)\,|s \in \Pi, q \in F\}$ is the set of accepting states, $W_p((s,q),(s',q')) = W(s,s')$ is a map assigning a positive weight to each transition. The product

Download English Version:

https://daneshyari.com/en/article/710438

Download Persian Version:

https://daneshyari.com/article/710438

Daneshyari.com