Thesis Proposal

Garrett Thomas

April 3, 2017

1 Background

In this thesis, we will study robot path planning, specifically discrete path planning in which the desired robot tasks are defined using temporal logics. Temporal logics are especially suited for defining robot tasks because of their ability to express not only fomulas constructed of atomic propositions and standard boolean connectives (conjunction, disjunction, and negation), but also temporal specifications e.g. ϕ is true at some point of time [1]. The particualar temporal logic we will be using is known as linear temporal logic (LTL). LTL formulas are defined according to the following grammar:

$$\phi ::= \phi_1 | \neg\phi_1 | \phi_1 \vee \phi_2 | \phi_1 U \phi_2$$

where ϕ_1 and ϕ_2 are LTL fomulas, $\neg$ and $\vee$ denote denote negation and disjunction respectively, and U is the temporal operator Until, with $\phi_1 U \phi_2$ meaning ϕ_1 is true until ϕ_2 becomes true. Given these operators, we can define the following additional prepositional operators:

$$\begin{aligned} \text{Conjunction: } & \phi_1 \wedge \phi_2 = \neg(\neg\phi_1 \vee \neg\phi_2) \\ \text{Implication: } & \phi_1 \Rightarrow \phi_2 = \neg\phi_1 \vee \phi_2 \\ \text{Equivalence: } & \phi_1 \Leftrightarrow \phi_2 = (\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1) \end{aligned}$$

We are also able to derive the following additional temporal operators:

$$\begin{aligned} \text{Eventuality: } & \diamond \phi_1 = TU\phi_1 \\ \text{Always: } & \Box\phi_1 = \neg \diamond \neg\phi_1 \end{aligned}$$

The approach of expressing desired specifications as temporal logics was originally designed for model checking, and provides a systematic and exhaustive method to check if a model satisfies some specification. In [3], Fainekos, G. et. all show how one can compute a discrete robot path which is guaranteed to satisfy the specifications using popular model checking tools to solve the dual problem i.e. finding a counter example to the negation of the desired specification.

2 Problem

In [2] Fainekos, G. et all describe an algorithm for finding a path that satisfies a given LTL formula.

1. Partition the environment
2. Represent this partition as a Finite Transition System (FTS)

3. Express the desired robot specifications as an LTL formula
4. Convert the LTL formula into Büchi automata
5. Construct the product automaton of the two calculated Büchi automata
6. Convert the product automaton into a directed graph
7. Find the path with the least amount of transitions to each accepting state using a breadth first search
8. Perform another breadth first search starting from each accepting state to find the shortest path back to itself
9. Return the accepting path with the least amount of transitions

A quick note explaining why we must perform the second breadth first search to find the path back to the accepting node and the existence of a path that satisfies the specification: Büchi automata accept infinite runs and an accepting run is a run that passes through an accepting state infinitely many times. Thus, for an accepting run to exist we must have a accepting state that is part of a Strongly Connected Component in the graph. The main problem with this method is the state explosion that can occur when creating the product automaton, even when the FTS and LTL formula are not particularly complex. The size of this resulting product automaton has the obvious detrimental effect of increased computation time of the breadth first searches. In this thesis we explore possible solutions to this state explosion problem. We wish to investigate

1. modelling the product automaton as a hybrid control system given the framework in [4]. Given this model, we will investigate the applicability of various results from the literature about control Lyapunov functions (CLF) and the existence of stabilizing feedback controllers.
2. finding/defining a measure which quantifies the distance each node is from an SCC containing an accepting state.

3 Motivation and Possible Ideas for the Distance Measure

The set of accepting states of the product automaton is $Q' \times F$ where Q' is the set of accepting states of the Büchi automaton corresponding to the FTS and F is the set of accepting states of the Büchi automaton corresponding to the LTL formula. Every state in the FTS Büchi automaton is and accepting state [1], thus to find an accepting state in the product automaton is essentially finding an accepting state of the LTL Büchi automaton. We therefore suggest assigning a distance measure in the LTL Büchi automaton and seeing if this measure is able to carry over to the product automaton. We will also examine what the product automaton looks like given different LTL specifications and explore possible measures when we restrict the specifications to a subset of LTL formulas. For example, limiting the specifications to formulas only expressing reachability while avoiding regions i.e. $\neg(\pi_1 \vee \pi_2 \vee \dots \pi_n) U \pi_{n+1}$ and sequencing i.e. $\diamond(\pi_1 \wedge \diamond(\pi_2 \wedge \diamond\pi_3))$. Specifications of these types may be easier to deal with because the robots task is finite in the sense that once it hits a certain state it can stay in that state forever and the runs remains accepting e.g. once it hits π_{n+1} in reachability while avoiding regions it can stay there and the run remains accepting.

4 References

References

[1] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT press, 1999.

[2] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[3] Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for mobile robots. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2020–2025. IEEE, 2005.