Linear Temporal Logic for Robot Path Planning

DE FILIPPIS Stefano

SAPIENZA
UNIVERSITÀ DI ROMA

Introduction

- **Problem:** move robotic agent on a plane from a starting position to a goal while avoiding obstacles or following some criteria

- **General Solution:** use Geometric A*

grid constraints movements

occluded areas approximated badly

- **Alternative:** use LTL where the planning problem consists in satisfying a suitable formula using a transition model for the robot

Problem Formulation

- We consider a robot moving in a polygonal environment P

$$\dot{x} = u(t) \quad x(t) \in P \subseteq \mathbb{R}^2$$

- The goal is to construct a control input so that the resulting trajectory x(t) satisfies a LTL formula built from a finite number of atomic propositions which label areas of interest in the environment

$$\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$$

- given an association map

$$h_C : P \to \Pi$$

SAPIENZA
UNIVERSITÀ DI ROMA

LTL

LTL is obtained from standard propositional logic by adding temporal operators such as eventually ($\Diamond$), always ($\Box$), next ($\bigcirc$) and until ($\mathcal{U}$).

Some interesting formulas can be expressed as:

- Reach goal while avoiding obstacles: $\neg(o_1 \vee o_2 \vee \ldots \vee o_n)\mathcal{U}\pi$
- Sequencing: $\Diamond(\pi_i \wedge \Diamond(\pi_j \wedge (\ldots\Diamond(\pi_k \wedge \Diamond\pi_l))))$
- Coverage: $\Diamond\pi_i \wedge \Diamond\pi_j \wedge \ldots \wedge \Diamond\pi_k$

LTL Motion Planning

In order to solve the previously defined problem we need to execute the 2 following steps:

1) Discrete Abstraction of Robot Motion

2) Temporal Logic Planning using Model Checking

Discrete Abstraction of Robot Motion

- Use Delaunay Triangulation to partition the workspace P

- Define $T : P \to Q$ as the map from robot continuous state to an equivalence class of the partition q_i

- Then we can abstract the robot motion by defining a finite transition system

$$D = (Q, q(0) \to_D, h_D)$$

- Whose dynamics are captured by the transition relation $q_i \to_D q_j$

Discrete Semantic

- we can define trajectories *p* of *D* as sequences of the form $p[i] = p_i \rightarrow_D p_{i+1} \rightarrow_D p_{i+2} \rightarrow_D \ldots$

- So, lift the problem formulation from the continuous to the discrete domain by recursively define the semantics of any path formula as:

  - $p[i] \models_D \pi$ iff $h_D(p(i)) = \pi$
  - $p[i] \models_D \neg\phi$ if $p[i] \not\models_D \phi$
  - $p[i] \models_D \phi_1 \vee \phi_2$ if $p[i] \models_D \phi_1$ or $p[i] \models_D \phi_2$
  - $p[i] \models_D \phi_1 \mathcal{U} \phi_2$ if there exists $j \geq i$ s.t. $p[j] \models_D \phi_2$, and for all j' with $i \leq j' < j$ we have $p[j'] \models_D \phi_1$

SAPIENZA UNIVERSITÀ DI ROMA

Temporal Logic Planning using Model Checking

- we are looking for computation paths *p[i]* that satisfy the temporal formula $p[0] \models_D \phi$

$\Downarrow$

generation of a trace of witnesses that satisfy the formula

- Problem: NuSMV does not support generation of witnesses
- Solution: try to solve the dual problem $p[0] \models_D \neg\phi$

- If initial formula is satisfiable then its dual problem is not and counterexample are generated

Implementation

- In MATLAB:

1) Create environment by defining the map and rooms vertices

2) Compute triangulation and obtain the dual graph to build the discrete transition system

3) Automatically create the main file for NuSMV

- in NuSMV

4) Launch the solver and save on a file the generated counterexample

- In MATLAB

5) Reconstruct the path by connecting the centers of passed triangles and do post smoothing

```
MODULE main
 VAR
  pi1 : boolean;
  pi2 : boolean;
  pi3 : boolean;
  pi4 : boolean;
  robot_1 : process robot(4);
 ASSIGN
  init(pi1) := TRUE;
  init(pi2) := FALSE;
  init(pi3) := FALSE;
  init(pi4) := FALSE;
  next(pi1) := case
            robot_1.state = 1 | robot_1.state = 4 : TRUE;
            TRUE : FALSE;
          esac;
  next(pi2) := case
            robot_1.state = 19 | robot_1.state = 21 : TRUE;
            TRUE : FALSE;
          esac;
  next(pi3) := case
            robot_1.state = 8 : TRUE;
            TRUE : FALSE;
          esac;
  next(pi4) := case
            robot_1.state = 25 | robot_1.state = 27 : TRUE;
            TRUE : FALSE;
          esac;

 LTLSPEC ! ( F ( pi2 & F ( pi3 & F ( pi4 & ( ! pi2 & ! pi3 ) U pi1 ) ) ) )
```

```
VAR
 state: 1 .. 32;
ASSIGN
 init(state) := 4;
 next(state) := case
            state = 1 : {11, 10, 4};
            state = 2 : {8, 10, 7};
            state = 3 : {15, 17, 19};
            state = 4 : {1, 17, 16};
            state = 5 : {26, 8, 9};
            state = 6 : {9, 8, 12};
            state = 7 : {11, 12, 2};
            state = 8 : {5, 2, 6};
            state = 9 : {5, 6, 28};
            state = 10 : {2, 13, 1};
            state = 11 : {7, 1, 16};
            state = 12 : {7, 6};
```

Post Smoothing

Algorithm 1 Post Smoothing

```
procedure Post Smoothing
    for each valid_state s do
        valid_successor ← successor(s)
        safe ← True
        stop ← False
        for each successor(s) next and safe not False and stop not True do
            line ← segment(s,next)
            for ecah room π do
                inter ← intersection(line,π)
                if inter not 0 and s and next not in π then
                    safe ← False
                if inter not 0 and next in π then
                    stop ← True
            if safe then
                valid_successor ← next
                s ← valid_successor
```

- For each current state skip successors when next states are in line of sight
- If starting state inside intersecting room continue
- If current successor inside the intersecting room still valid but stop searching
- Else intersection with obstacle valid successor is the previous one

Experiments

- First Formula:

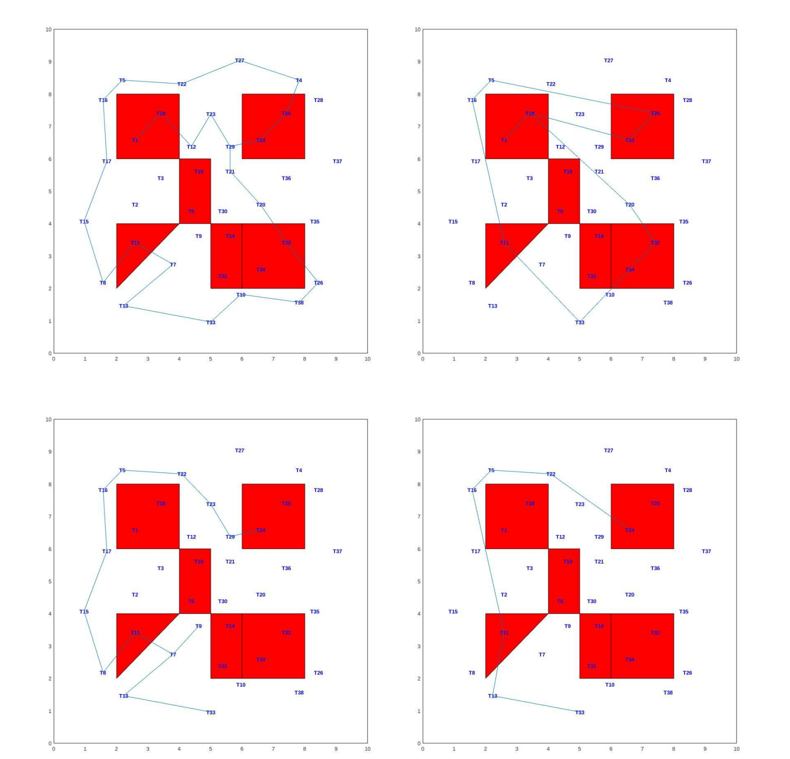
$$\Diamond(\pi_2 \wedge \Diamond(\pi_3 \wedge \Diamond(\pi_4 \wedge (\neg\pi_2 \wedge \neg\pi_3)\mathcal{U}\pi_1)))$$

- Second Formula

$$\bigcirc(\neg\pi_5 \wedge \neg\pi_6) \wedge \Diamond(\pi_4 \wedge \Diamond(\pi_3 \wedge \Diamond(\pi_2 \wedge \Diamond\pi_1)))$$

- Third Formula

$$(\neg\pi_1 \wedge \neg\pi_4 \wedge \neg\pi_5 \wedge \neg\pi_6)\mathcal{U}\pi_3$$



SAPIENZA UNIVERSITÀ DI ROMA