Temporal logic motion planning for mobile robots

Georgios E. Fainekos, Hadas Kress-Gazit and George J. Pappas

GRASP Laboratory
Departments of CIS and ESE University of Pennsylvania

{fainekos,hadaskg,pappasg}@grasp.upenn.edu

http://www.seas.upenn.edu/~fainekos/
Why temporal logic for motion planning?

How do we navigate a robot (even with simple dynamics) in a (complicated) environment???

Continuous
✓ Design Controller
✓ Verify

Discrete
✓ Discretize environment
✓ Ignore robot dynamics

Complex dynamics but ... NO complicated environments!
Complicated world but ... Can robot do it??

Can we combine the two approaches? Yes

Spatial and temporal specifications?
Use Temporal Logics
Control and computer science


Planning as model checking
[F. Giunchiglia and P. Traverso, '99]
[G. D. Giacomo and M. Y. Vardi, '99]
[F. Bacchus and F. Kabanza, '00]

Affine dynamical systems on simplexes
[L.C.G.J.M. Habets and J.H. van Schuppen., '04]
[C. Belta and L.C.G.J.M. Habets, '04]

Navigation functions
[E. Rimon and D. E. Kodischek, '92]
A simple example to guide us through …

Consider a robot that is moving in a square environment with four areas of interest denoted by $\pi_1$, $\pi_2$, $\pi_3$ and $\pi_4$. Initially, the robot is placed somewhere in the region labeled by $\pi_1$. The desired Specification for the robot given in natural language is: “Visit area $\pi_2$ then area $\pi_3$ then area $\pi_4$ and, finally, return to region $\pi_1$ while avoiding areas $\pi_2$ and $\pi_3$.”

**Input 2:** “Visit area $\pi_2$ then area $\pi_3$ then area $\pi_4$ and, finally, return to region $\pi_1$ while avoiding areas $\pi_2$ and $\pi_3$.”

**Output:** A hybrid controller that satisfies the specification by construction.
What can we express with temporal logics?

**Go to goal (reachability)**
\[ \varphi = \lozenge \pi_2 \]

**Coverage**
\[ \varphi = \lozenge \pi_2 \land \lozenge \pi_3 \land \lozenge \pi_4 \]

**Sequencing**
\[ \varphi = \lozenge (\pi_2 \land \lozenge \pi_3) \]

**Reachability with avoidance**
\[ \varphi = \neg (\pi_2 \lor \pi_3) U \pi_4 \]

**Recurrent Sequencing**
\[ \varphi = [] \lozenge (\pi_2 \land \lozenge \pi_3) \]

The simple example: “Visit area \( \pi_2 \) then area \( \pi_3 \) then area \( \pi_4 \) and, finally, return to region \( \pi_1 \) while avoiding areas \( \pi_2 \) and \( \pi_3 \)”

\[ \varphi = \lozenge (\pi_2 \land \lozenge (\pi_3 \land \lozenge (\pi_4 \land (\neg \pi_2 \land \neg \pi_3) U \pi_1))) \]
**Problem formulation**

**Model:** We consider a fully actuated, planar model of robot motion operating in a polygonal environment $P$. The motion of the robot is expressed as:

\[
\dot{x}(t) = u(t) \quad x(t) \in P \subseteq R^2 \quad u(t) \in U \subseteq R^2
\]

**Specification:** A linear temporal logic (LTL$_\mathcal{X}$) formula $\varphi$ that captures the robots’ desired behavior.

**Problem:** Given robot model, environment $P$, initial condition $x(0)$, and an LTL$_\mathcal{X}$ temporal logic formula $\varphi$, find control input $u(t)$ such that $x(t)$ satisfies $\varphi$. 
Overview of the Algorithm for Open-Loop Temporal Logic Motion Planning

Input 1: Polygonal Environment P

Input 2: Specification In Natural Language

Linear Temporal Logic

Model Checker (SPIN or NuSMV)

“Counter-example” discrete trail

Hybrid Controller

Continuous Implementation


Input 3: Robot model
A transition system
\[ D = (Q, q_0, \rightarrow, \Pi, h_D) \]
consists of
- A set of states \( Q \)
- An initial state \( q_0 \in Q \)
- The transition relation \( q_i \rightarrow q_j \)
- A set of observations \( \Pi \)
- The observation map \( h_D(q_i) = \pi_k \)

The language \( L(D) \) of \( D \) is the set of all the sequences of observations
i.e. \( s = \pi_0 \pi_1 \pi_2 \pi_0 \in L(D) \)
Discrete abstraction by triangulation

Partition the environment, Obtain discrete abstraction (FTS)
Ensure that triangulation preserves regions of interest
Overview of the Algorithm for the Open-Loop Temporal Logic Motion Planning

Input 1: Polygonal Environment $P$

Triangulation & Finite Transition Sys.

Model Checker (SPIN or NuSMV)

Hybrid Controller

Continuous Implementation

Input 2: Specification In Natural Language

Linear Temporal Logic

“Counter-example” discrete trail


The propositional formulas are formed using the traditional operators of conjunction ($\land$), disjunction ($\lor$), negation ($\neg$), implication ($\Rightarrow$), and equivalence ($\Leftrightarrow$). LTL formulas are obtained from the standard propositional logic by adding temporal operators such as eventually ($\Diamond$), always ($\Box$), and until ($U$).

<table>
<thead>
<tr>
<th>Informally</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventually $\pi_2$</td>
<td>$\Diamond \pi_2$</td>
</tr>
<tr>
<td>$\pi_0 \pi_1 \pi_2$</td>
<td></td>
</tr>
<tr>
<td>Eventually Always $\pi_1$</td>
<td>$\Diamond \Box \pi_1$</td>
</tr>
<tr>
<td>$\pi_0 \pi_0 \pi_1 \pi_1 \pi_1 \pi_1 \ldots$</td>
<td></td>
</tr>
<tr>
<td>$\pi_0$ until $\pi_2$</td>
<td>$\pi_0 U \pi_2$</td>
</tr>
<tr>
<td>$\pi_0 \pi_0 \pi_2$</td>
<td></td>
</tr>
</tbody>
</table>

LTL expresses temporal specifications along sequences of states.
Linear Temporal Logic LTL-\(X\)
(formally - continuous semantics)

The input LTL-\(X\) formulas are interpreted over continuous mobile robot trajectories. \(x[t]\) denotes the flow of \(x(s)\) under the input \(u(s)\) for \(t \leq s\). Proposition \(\pi \in \Pi\) represents an area of interest in the environment which can be characterized by a convex set of the form:

\[
P_i = \{ x \in \mathbb{R}^2 \mid \bigwedge a_k^T x + b_k \leq 0, \ a_k \in \mathbb{R}^2, b_k \in \mathbb{R}\}
\]

\[
x[t] \models \pi \iff h_C(x(t)) = \pi
\]
\[
x[t] \models \neg \varphi_1 \iff x[t] \not\models \varphi_1
\]
\[
x[t] \models \varphi_1 \lor \varphi_2 \iff x[t] \models \varphi_1 \ \text{or} \ \ x[t] \models \varphi_2
\]
\[
x[t] \models \varphi_1 U \varphi_2 \iff \exists s \geq t \ x[s] \models \varphi_2 \ \text{and} \ \ \forall t \leq t' < s \ x[t'] \models \varphi_1
\]

A trajectory \(x\) satisfies the specification \(\varphi\) and we write:

\[
x \models \varphi \iff x[0] \models \varphi
\]

where \(h_c\) is a function that maps the current state of the robot trajectory to an atomic proposition in the set \(\Pi\), i.e. \(h_c : \mathcal{P} \to \Pi\)
Overview of the Algorithm for the Open-Loop Temporal Logic Motion Planning

Input 1: Polygonal Environment $P$

Triangulation & Finite Transition Sys.

Model Checker (SPIN or NuSMV)

"Counter-example" discrete trail

Hybrid Controller

Continuous Implementation

Input 2: Specification In Natural Language

Linear Temporal Logic
Model checking is the algorithmic procedure for testing whether a specification formula holds over some semantic model. The model of the system is usually given in the form of a finite transition system. The specification formula is usually in the form of the temporal logics LTL or CTL.

Model checking problem:
\[ \forall p, p \models \varphi \]

Planning problem:
\[ \exists p, p \models \varphi \]

Generate discrete trajectory, originating at the initial condition, satisfying the temporal formula \( \varphi \), using model checking tools.

**SPIN:** [http://spinroot.com/spin/whatispin.html](http://spinroot.com/spin/whatispin.html)
LTL model checking
Automata theoretic approaches

**NuSMV:** [http://nusmv.irst.itc.it/](http://nusmv.irst.itc.it/)
CTL (and LTL) model checking
Symbolic based (BDD) approaches
NuSMV Model

\[ \varphi = \neg \diamond (\pi_2 \land \diamond (\pi_3 \land \diamond (\pi_4 \land (\neg \pi_2 \land \neg \pi_3) U \pi_1))) \]

\[ \psi = \bigwedge (\pi_2) \land \bigwedge (\pi_3) \land \bigwedge (\pi_4) \land (\neg \pi_2 \land \neg \pi_3) U \pi_1) \]

\[
\text{The trajectory generated by NuSMV, satisfying this formula is: 33, 34, 24, 25, 27, 16, 15, 14, 3, 4, 5, 32, 23, 26, 29, 30, 3, 14, 33}
\]
Overview of the Algorithm for the Open-Loop Temporal Logic Motion Planning

Input 1: Polygonal Environment P

Triangulation & Finite Transition Sys.

Input 2: Specification In Natural Language

Linear Temporal Logic

Model Checker (SPIN or NuSMV)

“Counter-example” discrete trail

Hybrid Controller

Continuous Implementation


Hybrid Controller Implementation

Is the partition (triangulation) consistent with dynamics?
Yes, if partition is a **bi-simulation**.

A triangulation is a **bisimulation** if the system can move between any two adjacent triangles regardless of the initial state. For each triangle, we design three controllers ensuring that system exits the triangle from the desired facet to the adjacent triangle.

**Thm:** There exist (many) affine vector fields
\[
\dot{x}_P = u_P \quad u_P = Ax + b \in U_P
\]
on any triangle, satisfying the bisimulation property.

Affine functions on simplexes are uniquely defined on vertices. The set of all controllers can be parameterized by the values on the vertices.
Based on the discrete path, we design bi-simulation controllers driving the robot from one triangle to the adjacent triangle. We can take advantage of the non-unique affine solutions by matching affine vector fields on common facets, (if possible).
Overview of the Algorithm for the Open-Loop Temporal Logic Motion Planning

Input 1: Polygonal Environment P

Triangulation & Finite Transition Sys.

Model Checker (SPIN or NuSMV)

"Counter-example" discrete trail

Hybrid Controller

Input 2: Specification In Natural Language

Linear Temporal Logic

Continuous Implementation
Main Result - Completeness

If the robot is modeled as $\dot{x}(t) = u(t)$ and $\delta(0, \epsilon) \in U$ then

$$x \models C\varphi \text{ iff } p \models D\varphi$$

If the system is modeled as $\dot{x} = Ax + Bu$ and conditions (*) then

$$x \models C\varphi \text{ iff } p \models D\varphi$$

In any case: $x \models C\varphi \Rightarrow p \models D\varphi$

For the multi-robot case using the above dynamics: $p \models D\varphi \not\Rightarrow x \models C\varphi$

Other dynamics: $p \models D\varphi \Rightarrow x \models C\varphi$

Continuous refinement

Start

Goal
Examples

Spec: Go to areas 1, 2, 3, 4, 5, 6 in no particular order.

Computation time: Triangulation < 1sec, Model checking < 1sec, Hybrid Controller ~13sec (MATLAB)
Examples

Spec: Go to area 2, then to area 1 and then cover areas 3, 4, 5 - all this, while avoiding obstacles $O_1, O_2, O_3$
Larger examples

Spec: Go to the two black rooms

Problem Size
1156 observables
9250 triangles
Solution path: 145 triangles
145 controllers

Computation time
- Triangulation: A few seconds
- NuSMV: 55 seconds
- Matlab: 90 seconds
Conclusions

- Presented a **formal and compact** way to capture complicated path planning specifications. Also, a unified specification language for many tasks.

- A connection between high level **AI planning** and low level **controller design**

- **Completeness** results (for certain cases)

- Computationally **efficient** approach

- **Robustness** with respect to the initial conditions within a class of the partition
Future Issues

More general decompositions/abstractions
  Abstraction should depend on more complicated dynamics
  Robust abstractions with respect to modeling/sensing noise

Open-loop versus closed loop planning
  Robust satisfaction of temporal formulas
  Feedback plans will result in hybrid controllers

Multi-robot logics
  Composition semantics for multiple robots

From natural language to robot motion

Experiments (ground vehicles and UAVs)
Thank You!

Questions?

http://www.seas.upenn.edu/~fainekos/