Temporal logic motion planning for mobile robots

G. E. Fainekos, H. Kress-Gazit and G. J. Pappas

GRASP Laboratory,
Departments of CIS and ESE,
University of Pennsylvania.

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

Temporal logic motion planning for mobile robots
[G. E. Fainekos, H. Kress-Gazit and G. J. Pappas '05]

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

Input 1 (Environment)

Input 2 (Specification):
“Visit area $a_2$ then area $a_3$ then area $a_4$ and, finally, return to region $a_1$ while avoiding areas $a_2$ and $a_3$”

Input 3 (Robot position)

Output: A hybrid controller for the mobile robot that satisfies the specification by construction.
How about multi-robot navigation?

**Input 1 (Environment):**

- $\alpha_1$
- $\alpha_2$
- $\alpha_3$
- $\alpha_4$

**Input 2 (Specification):**

“Robot A and B cover areas $\alpha_1$ and $\alpha_3$ and if Robot A visits area $\alpha_3$ then it should also go to area $\alpha_4$”

**Input 3**

(Robots’ or groups of robots’ positions)

**Output:** A hybrid controller for each mobile robot that satisfies the specification by construction.
What can we express with temporal logics? (Single robot case)

Let the **atomic propositions** be $\pi_j$, where $j$ is the area of interest $a_j$.

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

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

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

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

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

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 $\pi_3$”
\[
\varphi = \Diamond (\pi_2 \land \Diamond (\pi_3 \land \Diamond (\pi_4 \land \neg \pi_2 \land \neg \pi_3) \lor \pi_1)))
\]
What can we express with temporal logics? (Multi robot case)

Let the **atomic propositions** be $p_{ij}$, here $i$ is the robot (or group of robots) and $j$ be the area of interest $\alpha_j$.

**Go to goal (reachability)**

$\phi = \Diamond \pi_A \land \Diamond \pi_B$

**Coverage**

$\phi = \Diamond (\pi_A \lor \pi_B) \land \Diamond (\pi_A \lor \pi_B) \land \Diamond (\pi_A \lor \pi_B)$

**Sequencing**

$\phi = \Diamond (\pi_A \land \Diamond (\pi_A \lor \pi_B))$

**The simple example:** “Robot A and B cover areas $\alpha_1$ and $\alpha_3$ and if Robot A visits area $\alpha_3$ then it should also go to area $\alpha_4$”

$\phi = \Diamond (\pi_A \lor \pi_B) \land \Diamond (\pi_A \lor \pi_B) \land (\Diamond \pi_A \rightarrow \Diamond \pi_A)$
Problem formulation

**Model:** We consider $n$ fully actuated mobile robots operating in a planar polygonal environment $P$. The motion of each robot $i$ is expressed as:

$$\frac{dx_i}{dt} = u_i(t) \quad \text{with} \quad x_i(t) \in P \subseteq \mathbb{R}^2 \quad \text{and} \quad u_i(t) \in U_i \subseteq \mathbb{R}^2$$

and for all the robots:

$$\frac{dX}{dt} = U(t) \quad \text{with} \quad X(t) = [x_1(t),...,x_n(t)]^T \quad \text{and} \quad U(t) = [u_1(t),...,u_n(t)]^T$$

**Specification:** A linear temporal logic (LTL$_X$) formula $\varphi$ that captures the robots’ desired behavior.

**Problem:** Given $n$ robot models, an environment $P$, initial conditions $X(0)$, and an LTL$_X$ temporal logic formula $\varphi$, find a control input $U(t)$ such that $X(t)$ satisfies $\varphi$. 
Overview of the Algorithm (1)

**Input 1:** Polygonal Environment $P$

**Input 2:** Specification In Natural Language


**Input 3:** Robot model

1. Triangulation & Finite Transition Sys.
2. Linear Temporal Logic
3. Black Box

“Counter-example” discrete trail

Model Checker (SPIN or NuSMV)

Hybrid Controller

Continuous Implementation
Finite Transition Systems (FTS)

- A transition system for robot $i$
  
  $D_i = (Q, q_0, \rightarrow, \Pi, h_i)$

  consists of
  
  A set of states $Q$
  An initial state $q_0 \in Q$
  The transition relation $q_j \rightarrow q_j'$
  A set of observations $\Pi$
  The observation map $h_i(q_j) = \pi_{ik}$

- The language $L(D_i)$ of $D_i$ is the set of all the sequences of observations i.e. $p = \pi_{i0}\pi_{i1}\pi_{i2}\pi_{i0} \in L(D_i)$
Discrete abstraction by triangulation

- Partition the environment, Obtain discrete abstraction
- Ensure that triangulation preserves regions of interest
Overview of the Algorithm (2)

**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

---


Linear Temporal Logic LTL$_X$ (informally – discrete semantics)

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 $\pi_{i2}$</td>
<td>$\Diamond \pi_{i2}$</td>
</tr>
<tr>
<td>Eventually Always $\pi_{i1}$</td>
<td>$\Diamond \Box \pi_{i1}$</td>
</tr>
<tr>
<td>$\pi_{i0}$ until $\pi_{i2}$</td>
<td>$\pi_{i0} \ U \pi_{i2}$</td>
</tr>
<tr>
<td>$\pi_{i0}\pi_{i1}\pi_{i2}$</td>
<td>$\pi_{i0}\pi_{i1}\pi_{i2}$</td>
</tr>
</tbody>
</table>
The input \( \text{LTL}_{-X} \) formulas are interpreted over \textbf{continuous} mobile robot trajectories. \( X[t] \) denotes the flow of \( X(s) \) under the inputs \( U(s) \) for \( t \leq s \).

- A \textbf{proposition} \( \pi_{ij} \in \Pi \) represents an area of interest in the environment which can be characterized by a convex set of the form:

\[
P_{ij} = \{ x \in R^2 \mid \sum_k a_{ijk}^T x + b_{ijk} \leq 0, a_{ijk} \in R^2, b_{ijk} \in R \}
\]

- \( X[t] \models \pi \) 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 \( X[t] \models \varphi_1 \) or \( X[t] \models \varphi_2 \)
- \( X[t] \models \varphi_1 U \varphi_2 \) iff
  \[
  \exists s \geq t \ X[s] \models \varphi_2 \quad \text{and} \quad \forall t \leq t' < s \ X[t'] \models \varphi_1
  \]

When a set of trajectories \( X \) satisfies the specification \( \varphi \), we write:

\[
X \models \phi \quad \text{iff} \quad X[0] \models \varphi
\]

where \( h_C \) is a function that maps the current state of the robot trajectories to a set of atomic propositions in \( \Pi \), i.e. \( h_C : P \rightarrow 2^\Pi \).
Overview of the Algorithm (3)

**Input 1:** Polygonal Environment P

1. Triangulation & Finite Transition Sys.

2. Model Checker (SPIN or NuSMV)

   "Counter-example" discrete trail

3. Hybrid Controller

4. Continuous Implementation

**Input 2:** Specification In Natural Language

1. Linear Temporal Logic

2. Hybrid Controller
Planning via Model Checking

**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

\[ \exists p, p \models \varphi \]

Dual Problems

Model Checking problem

\[ \forall p, p \models \neg \varphi \]

Generate discrete trajectory, originating at the initial condition, satisfying the temporal formula \( \varphi \), using model checking tools (i.e. SPIN or NuSMV).
Example: NuSMV Model

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

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

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 (4)

**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

---

*C. Belta, V. Isler, and G. J. Pappas,* "Discrete abstractions for robot motion planning and control," IEEE Transactions on Robotics, Accepted for publication.

Hybrid Controller Implementation

When is the partition (triangulation) consistent with the dynamics? If it is a **bi-simulation**.

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

**Thm:** There exist (many) affine vector fields

\[
\frac{dx_p}{dt} = u_p \quad u_p = Ax + b \in U_p
\]

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.
Refinement

- 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 (5)

**Input 1:** Polygonal Environment $P$

Triangulation & Finite Transition Sys.

Model Checker (SPIN or NuSMV)

Linear Temporal Logic

“Counter-example” discrete trail

Hybrid Controller

Continuous Implementation

**Input 2:** Specification In Natural Language
Main Result – Completeness

Single Robot case

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

$$x|_0 = \mathcal{C}\varphi \iff p|_0 = \mathcal{D}\varphi$$

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

$$x|_0 = \mathcal{C}\varphi \iff p|_0 = \mathcal{D}\varphi$$

Extensions to the main result

**Single Robot:**
In any case:  \( x|\models \mathcal{C} \varphi \Rightarrow p|\models \mathcal{D} \varphi \)

Other dynamics:  \( p|\models \mathcal{D} \varphi \Rightarrow x|\models \mathcal{C} \varphi \)

**Multi-Robot:**
With certain (*) composition semantics for the discrete system:
\[ X|\models \mathcal{C} \varphi \Rightarrow p|\models \mathcal{D} \varphi \]

For a particular fragment of LTL (*):
\[ p|\models \mathcal{D} \varphi \Rightarrow X|\models \mathcal{C} \varphi \]

For the full LTL (Synchronization primitives are required):
\[ p|\models \mathcal{D} \varphi \nRightarrow X|\models \mathcal{C} \varphi \]

(*) Unpublished results
Continuous refinement

- Star: Start
- Green: Goal
Example

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

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

- **Spec:** Go to area 2, then to area 1 and then cover areas 3, 4, 5 – all this, while avoiding obstacles O₁, O₂, O₃

---

**SPIN**

**NuSMV**
Larger Example

**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
Future Extensions

- Assume two groups of robots A (Red) and B (Green)
  - Due to the construction of the hybrid controller, the same controller applies to a number of robots in the same neighborhood

- **Spec:** Coverage of the areas of interest

  \[ \varphi = \Diamond (\pi_{A1} \lor \pi_{B1}) \land \Diamond (\pi_{A3} \lor \pi_{B3}) \land \Diamond (\pi_{A4} \lor \pi_{B4}) \]
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
Issues Under Investigation and Future Work

- 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 (unpublished results)
  - Hierarchical structure: Application of temporal logic specifications on the level of swarms or group of robots
  - Synchronization primitives for multi-robot applications
  - Collision avoidance resolution
- From natural language to robot motion
- Experiments (ground vehicles and UAVs)
Thank You!

- Questions?