Overview

- **Robot Task and Motion Planning** as Synthesis
- **Sensor-Based Planning** as Reactive Synthesis
- Synthesizing **Hybrid Controllers**
- **Variants**
  - Complex dynamics
  - Multi-robot missions
  - High-dimensional systems
- **Specification Languages and Tools for Human Users**
Classical Motion Planning
(Piano Mover’s Problem)
Generalizing this...
Linear Temporal Logic (LTL)

Syntax

• Boolean propositions $\pi$

• Boolean operators:
  
  $\land$ (and)  $\lor$ (or)  $\neg$ (not)  $\implies$ (implies)

• Temporal operators:
  
  $\mathcal{U}$ (until)  $\circ$ (next)  $\Box$ (always)  $\Diamond$ (eventually)

Semantics over infinite (discrete state) executions

- next $\circ \varphi$
- always $\Box \varphi$
- eventually $\Diamond \varphi$
Linear Temporal Logic (LTL)

Syntax
• Boolean propositions $\pi$
• Boolean operators:
  \[\land (\text{and}) \quad \lor (\text{or}) \quad \lnot (\text{not}) \quad \implies (\text{implies})\]
• Temporal operators:
  $\mathcal{U}$ (until) \quad $\bigcirc$ (next) \quad $\Box$ (always) \quad $\Diamond$ (eventually)

Semantics over infinite (discrete state) executions

- safety $\Box \varphi_{\text{safe}}$
- reachability $\Diamond \varphi_{\text{goal}}$
- liveness $\Box \Diamond \varphi_{\text{task}}$
- stability $\Diamond \Box \varphi_{\text{stable}}$
- response $\Box (\varphi_{\text{sense}} \rightarrow \bigcirc \varphi_{\text{react}})$
Temporal Logic Planning

$\Diamond A \land \Box \Diamond B \land \Box \Diamond C$
Temporal Logic Planning

“Always eventually deliver all packages and never crash.”

Specify requirements in a formal language

aka synthesis

Find correct-by-construction plan
Overview

- **Robot Task and Motion Planning** as Synthesis
- **Synthesizing Hybrid Controllers**
- **Sensor-Based Planning** as Reactive Synthesis
- **Variants**
  - Complex dynamics
  - High-dimensional systems
  - Multi-robot missions
- **Specification Languages and Tools for Human Users**
Hybrid Control Architecture

discrete logic + continuous dynamics

cyber-physical system

\[ \text{min} \int_{t_0}^{T} L(x, u) dt \]
\[ \text{s.t.} \quad \dot{x} = f(x, u) \]
\[ g(x, u) \leq 0 \]

Figure courtesy of R.M. Murray
Temporal Logic Planning for Cyber-Physical Systems

Create discrete abstraction → Synthesize correct discrete controller → Implement hybrid control on physical system

Dynamical system → Labeled transition system
Create discrete abstraction

Dynamical system → Labeled transition system

(Figure from BeltaIP04)

AlurHLP00, BeltaH06, HabetsCS06, Kress-GazitFP07, KloetzerB08, TabuadaP06, WongpiromsarnTM12,...
Create discrete abstraction

Dynamical system

Labeled transition system

(Figures from BhatiaKV10, KaramanF09, KaramanF12, PlakuH10,...)

BhatiaKV10, KaramanF09, KaramanF12, PlakuH10,...
Overview

• **Robot Task and Motion Planning** as Synthesis
• Synthesizing **Hybrid Controllers**
• **Sensor-Based Planning** as Reactive Synthesis
• **Variants**
  – Complex dynamics
  – High-dimensional systems
  – Multi-robot missions
• Specification Languages and Tools for **Human Users**
Most robotic systems are not closed

- React to potentially adversarial environments
- Safety-critical conditions
Sensor-Based High-Level Planning

Based on slides by H. Kress-Gazit
Reactive Synthesis

Given \( \varphi = \varphi_e \Rightarrow \varphi_s \)

- environment assumptions
- system guarantees

synthesize a strategy \( \mathcal{M} \) such that \( \mathcal{M} \models \varphi \)
GR(1): Tractable Reactive Synthesis

\[ \varphi = \varphi_e \quad \implies \quad \varphi_s \]

- Environment assumptions
- System guarantees

\[ \square \lozenge \varphi_e^g \quad \implies \quad \square \lozenge \varphi_s^g \]

Polynomial in size of state space

Curious? Pay attention during Roderick’s lecture on Thursday

[Bloem et al, ’12]
Sensor-Based High-Level Planning

Based on slides by H. Kress-Gazit

Sensor inputs → Actions → Binary Propositions → Discrete Abstraction → Known workspace

Additional notes:
- Sensor inputs
- Actions
- Binary Propositions: $\varphi = \varphi_e \implies \varphi_s$
- Finite State Transducer
- Hybrid Controller
- Correct robot motion and action

Contact: vasu@caltech.edu -- ExCAPE Summer School 2015
A simple example

Actions: \( r_1, r_2, \text{camera} \)

Sensors: \( \text{person} \)

“Robot starts in region \( r_1 \) with the camera off”
\[
q_0 \quad \neg \text{person} \quad \neg \text{camera}
\]

“Activate the camera if and only if a person is seen”
\[
\neg \text{camera} \quad \langle \neg \text{camera} \rangle \quad \langle \neg \text{person} \rangle \quad \langle \text{camera} \rangle \quad \neg \text{person} \\
\neg \text{person} \quad \langle \text{person} \rangle \quad \langle \text{camera} \rangle \quad \langle \neg \text{camera} \rangle \quad \langle \text{person} \rangle
\]

“Always eventually visit \( r_2 \)”
\[
\langle \Box \Diamond r_2 \rangle
\]

vasu@caltech.edu -- ExCAPE Summer School 2015
Continuous-time Semantics

- Camera turns on
- Motion from r1 to r2

Synthesis assumes transitions are instantaneous!
Approach 1:

assume actions have similar duration
enforce simultaneous completion

When assumptions are violated:
- delayed response
- can be confusing
Approach II: assume actions have similar duration enforce simultaneous *initiation*

When assumptions are violated:
- can end up in an unmodeled state
Approach III:
Explicit Continuous Execution Semantics

- Explicitly model action initiation/completion

**Actions:** $r_1$, $r_2$, camera  
**Sensors:** person, $c_{r_1}$, $c_{r_2}$, $c_{camera}$

- Specification (excerpt)

  “Robot starts in region $r_1$ with camera
  $c_{r_1} \land \neg c_{camera}$

  “Activate the camera if and only if you see a person”
  $\land \square (\lozenge \text{person} \Leftrightarrow \lozenge \text{camera})$

  “Visit $r_2$”
  $\land \lozenge c_{r_2}$

  Additional Safety Conditions
  $\land \square ((c_{r_1} \land \lozenge r_2) \Rightarrow (\lozenge c_{r_1} \lor \lozenge c_{r_2}))$
  $\land \square ((c_{camera} \land \lozenge \text{camera}) \Rightarrow \lozenge \text{camera})$
Approach III:

no assumption on action duration
Explicitly model initiation/completion

Actions

<table>
<thead>
<tr>
<th>camera</th>
<th>r2</th>
</tr>
</thead>
</table>

Sensors

<table>
<thead>
<tr>
<th>c_{camera}</th>
<th>c_{r2}</th>
</tr>
</thead>
</table>

vasu@caltech.edu -- ExCAPE Summer School 2015
Trade-offs

Approach III

Approach I & II

Synthesis time:

42ms

16ms

“Timing Semantics for Abstraction and Execution of Synthesized High-Level Robot Control” [RamanPFK, TRO’15]
Overview

• **Robot Task and Motion Planning** as Synthesis
• **Sensor-Based Planning** as Reactive Synthesis
• Synthesizing **Hybrid Controllers**
• **Variants**
  – Complex dynamics
  – Multi-robot missions
  – High-dimensional systems
• Specification Languages and Tools for **Human Users**
Complex Dynamics
Create discrete abstraction

Synthesize correct discrete controller

Implement hybrid control on physical system

Dynamical system

Labeled transition system

vasu@caltech.edu -- ExCAPE Summer School 2015
Unicycle

\[ \dot{x} = u_s \cos \theta \]
\[ \dot{y} = u_s \sin \theta \]
\[ \dot{\theta} = u_\omega \]

Define \( \nu = \frac{r}{2} (u_r + u_l) \)
\[ \omega = \frac{r}{L} (u_r - u_l) \]

Then
\[ \dot{x} = \nu \cos \theta \]
\[ \dot{y} = \nu \sin \theta \]
\[ \dot{\theta} = \omega \]

“Planning Algorithms”, S. M. LaValle ‘06
Synthesizing Low-Level Control Post Hoc

• High-order nonlinear dynamics

• Produces:
  • Library of atomic controllers
  • Verified subset of phase space

• Continuous trajectory (top) faithfully executes the synthesized automaton (bottom)

Based on slides by J. A. DeCastro

“Guaranteeing Reactive High-Level Behaviors for Robots with Complex Dynamics” [DeCastroK, IROS’13]
Iterative Abstraction Adaptation Based on Dynamics

“Dynamics-Driven Adaptive Abstraction for Reactive High-Level Mission and Motion Planning” [DeCastroRK, ICRA’15]
Multi-Robot Missions
Household Cleaning Example

- Visit the living room and bedroom
- Avoid collisions

Based on slides by J. A. DeCastro
I -- Collision Avoidance Using a Local Planner

“The two Naos locally avoid each other

“Generalized, Collision-Free Reactive Mission and Motion Planning for Multi-Agent Systems”
[DeCastroARRK, under review ‘15]
Using a Local Planner

Assumptions on Deadlock Resolution

For each transition, either:
1. The environment never causes deadlock
2. The environment eventually stops causing deadlock
3. No restriction on the environment

Which assumptions we make depends on what our local planner can resolve!

Example: Deadlock should not occur when robot 1 is in the Hall moving toward the Living Room and has been blocked from entering the Bedroom.
II -- Negotiating Requirements

Alice’s Specification
-Alice starts from bridge and goes to groceryStore

Bob’s Specification
-Bob starts from groceryStore and goes to postOffice

“Let’s Talk: Autonomous Conflict Resolution for Robots Carrying out Individual High-level Tasks in a Shared Workspace”
II -- Negotiating Requirements

Alice’s Specification
-Alice starts from bridge and goes to groceryStore
-Bob will not be in her way

Bob’s Specification
-Bob starts from groceryStore and goes to postOffice
-Alice will not be in his way

“Let’s Talk: Autonomous Conflict Resolution for Robots Carrying out Individual High-level Tasks in a Shared Workspace”
II -- Negotiating Requirements

### Alice’s Specification
- Alice starts from bridge and goes to groceryStore
- Bob will not be in her way

### Bob’s Specification
- Bob starts from groceryStore and goes to postOffice
- Alice will not be in his way

“Let’s Talk: Autonomous Conflict Resolution for Robots Carrying out Individual High-level Tasks in a Shared Workspace”

II -- Negotiating Requirements

- Alice starts from bridge and goes to groceryStore
- Bob will not be in her way
- Bob is heading to postOffice
- Alice should try not to get into Bob’s way

- Bob starts from groceryStore and goes to postOffice
- Alice will not be in his way

Based on slides by K. Wong.
III -- Composing Motion Primitives

Application:
persistent surveillance with quadrotors

\[ \mathcal{A}(\square(\Diamond(rX\text{gather} \land (\Diamond rX\text{upload})))) \]
\[ \mathcal{E}(\square(\Diamond(rX\text{gather} \land (\Diamond rX\text{upload})))) \]

Based on slides by I. Saha
“Automated Composition of Motion Primitives for Multi-Robot Systems from Safe LTL Specifications” [SahaRKPS, IROS’14]
III -- Composing Motion Primitives

1. Pre-compute a set of control laws, called motion Primitives
2. Encode the planning problem as a bounded synthesis problem
3. Solve using an SMT solver (Z3)

Tool!
Complan
(COmpositional Motion PLANner)
http://www.seas.upenn.edu/~isaha/complan.shtml

Based on slides by I. Saha
“Automated Composition of Motion Primitives for Multi-Robot Systems from Safe LTL Specifications” [SahaRKPS, IROS’14]
High-Dimensional Systems
Create discrete abstraction → Synthesize correct discrete controller → Continuously implement discrete solution

Discrete abstraction is too expensive for high-dimensional systems (> 5 dim)
Optimization-Based Temporal Logic Planning

System Dynamics → Specification $\varphi$ → Trajectory Parametrization

Constrained Optimization Problem (e.g. MILP) → Solver

Optimal control input enforcing $\varphi$

[Karaman et al. 08, Kwon and Agha 08, Wolff et al. 14, Raman et al 14] solutions in ms for systems with 5+ continuous state variables!
Signal Temporal Logic (STL)

- **Continuous predicates:** $\mu(x) > 0$

- **Boolean operators:**
  - $\land$ (and)
  - $\lor$ (or)
  - $\neg$ (not)
  - $\implies$ (implies)

- **Temporal operations:**
  - $\forall U_{[a,b]} \psi$ (until)
  - $\square_{[a,b]} \varphi$ (always)
  - $\diamond_{[a,b]} \varphi$ (eventually)

No need to discretize the state space!

"At all times between a and b from now"
Quantitative Semantics for STL

Robustness function \( \rho^\varphi : \mathcal{X} \times \mathbb{N} \to \mathbb{R} \)

\[
(x, t) \models \varphi \equiv \rho^\varphi(x, t) > 0
\]

\[
|x'_t - x_t| < \rho^\varphi(x, t)
\Rightarrow (x', t) \models \varphi
\]
Open-Loop Synthesis

Problem: find a bounded-time sequence of inputs
• satisfying an STL formula
• minimizing some cost

Solution: Encode everything (including the formula) as a Mixed Integer Linear Program
### Encoding STL as MILP Constraints

Given $\varphi$, recursively generate constraints for $\psi \in cl(\varphi)$

<table>
<thead>
<tr>
<th>Boolean</th>
<th>Quantitative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduce</strong></td>
<td>$z_t^\psi$</td>
</tr>
<tr>
<td><strong>Constrained such that</strong></td>
<td>$z_t^\psi = 1 \iff (x, t) \models \psi$</td>
</tr>
<tr>
<td><strong>Enforce</strong></td>
<td>$z_0^\varphi = 1$</td>
</tr>
</tbody>
</table>

$(x, 0) \models \varphi \iff r_0^\varphi > 0$
Example: STL Encoding

\( (x, 0) \models \varphi \iff r_{0}^{\varphi} > 0 \)

\[ \varphi = \Box_{[0, 10]} \Diamond_{[0, 5]} ((T > 20) \land (T < 30)) \]

\[ r_{t}^{(T > 20)} = T - 20 \]

\[ r_{t}^{(T < 30)} = 30 - T \]

\[ r_{t}^{((T > 20) \land (T < 30))} = \min(r_{t}^{(T > 20)}, r_{t}^{(T < 30)}) \]

\[ r_{t}^{[0, 5]}^{((T > 20) \land (T < 30))} = \max_{i=0, \ldots, 5} (r_{i}^{((T > 20) \land (T < 30))}) \]

\[ r_{t}^{[0, 10]} \Diamond_{[0, 5]}^{((T > 20) \land (T < 30))} = \min_{i=0, \ldots, 10} (r_{i}^{[0, 5]}^{((T > 20) \land (T < 30))}) \]
Open-Loop Synthesis

Given:
Continuous-time system \( \dot{x} = f(x, u, w) \)
STL specification \( \varphi \)
Initial state \( x_0 \)
Cost function \( J \) on runs

Compute:
\[
\arg \min_u J(x(x_0, u), u)
\quad \text{s.t. } x(x_0, u) \models \varphi
\]
Reactive Synthesis

Given:
Continuous-time system \( \dot{x} = f(x, u, w) \)
STL specification \( \varphi = \varphi_e \implies \varphi_s \)
Initial state \( x_0 \)
Cost function \( J \) on runs

Compute:
\[
\arg\min_{u^N} \max_{w^N \in \{w \in W^N \mid w \models \varphi_e\}} J(\xi(x_0, u^N, w^N))
\]
\[
s.t. \quad \forall w^N \in W^N, \quad \xi(x_0, u^N, w^N) \models \varphi
\]

“Reactive Synthesis from Signal Temporal Logic Specifications” [RamanDSMS, HSCC’15]
CEGIS for STL Reactive Synthesis

Initial Candidate Disturbance(s) → OPENLOOP → Candidate Input → FALSIFY

Counterexample found
No counterexample

Finite Horizon Control Input

vasu@caltech.edu -- ExCAPE Summer School 2015
CEGIS for STL Reactive Synthesis

1: procedure CEGIS($\xi$, $x_0$, $N$, $\varphi$, $J$)
2: Let $\mathbf{w}^0 = (w_0^0, w_1^0, \ldots w_{N-1}^0)$, s.t. $\mathbf{w}^N \models \varphi_e$
3: $W_{cand} = \{\mathbf{w}^0\}$
4: while True do
5:    $\mathbf{u}^0 \leftarrow \arg\min_{\mathbf{u} \in U^N} \max_{\mathbf{w}^0 \in W_{cand}} (J(\xi(x_0, \mathbf{u}, \mathbf{w}^0)))$
6:    s.t. $\forall \mathbf{w}^0 \in W_{cand}$, $\xi(x_0, \mathbf{u}, \mathbf{w}^0) \models \varphi_s$,
7:    if $\mathbf{u}^0 == \text{null}$ then
8:        Return INFEASIBLE
9:    end if
10:   $\mathbf{w}^1 \leftarrow \arg\min_{\mathbf{w} \in W^N} \rho^\varphi(\xi(x_0, \mathbf{u}^0, \mathbf{w}), 0)$
11:    s.t. $\mathbf{w}^1 \models \varphi_e$
12:    if $\rho^\varphi(\xi(x_0, \mathbf{u}^0, \mathbf{w}^1)) > 0$ then
13:        Return $\mathbf{u}^0$
14:    else
15:        $W_{cand} \leftarrow W_{cand} \cup \{\mathbf{w}^1\}$
16:    end if
17: end while
18: end procedure

Initial Guess

OPENLOOP

FALSIFY

Inductive Step

vasu@caltech.edu -- ExCAPE Summer School 2015 53
Model Predictive Control

• So far: synthesize a finite trace

\[ x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_{N-1} \]

• MPC: do this repeatedly

\[ x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_H \rightarrow x_{H+1} \rightarrow x_{H+2} \]

“Model Predictive Control with Signal Temporal Logic Specifications” [RamanDMMSS, CDC’14]
Case Study: Autonomous Driving

\[
\begin{bmatrix}
\dot{x}_\text{ego} \\
\dot{y}_\text{ego} \\
\dot{v}_\text{ego}
\end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_\text{ego} \\
y_\text{ego} \\
v_\text{ego} \end{bmatrix} + \begin{bmatrix} 0 \\
0 \\
1 \end{bmatrix} \mathbf{u}
\]

\[
\begin{bmatrix}
\dot{x}_\text{adv} \\
\dot{y}_\text{adv} \\
\dot{v}_\text{adv}
\end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_\text{adv} \\
y_\text{adv} \\
v_\text{adv} \end{bmatrix} + \begin{bmatrix} 0 \\
0 \\
1 \end{bmatrix} \mathbf{w}
\]

\[\varphi_e = \square(|\mathbf{w} - \mathbf{w}^{\text{ref}}| < 0.1)\]

\[\varphi_s = \square(|y_t^{\text{ego}} - x_t^{\text{adv}}| < 2) \Rightarrow \square_{[0,2]}(|v_t^{\text{ego}}| < 0.1)\]

\[J(\xi(x_t, \mathbf{u}^H, \mathbf{w}^H)) = \sum_{l=0}^{H-1} |v_{t+l}^{\text{ego}} - 1|\]
Case Study: Autonomous Driving

![Graph showing distance and velocity over time for ego and adversary vehicles.]

- Distance of Ego vehicle from intersection
- Distance of Adversary vehicle from intersection

- Velocity of Ego Vehicle
- Velocity of Adversary Vehicle
Tool!

BluSTL: Controller Synthesis from Signal Temporal Logic Specifications *(presented at ARCH 2015)*

https://github.com/BluSTL/BluSTL

(BSD License)
Overview

- **Robot Task and Motion Planning** as Synthesis
- **Sensor-Based Planning** as Reactive Synthesis
- Synthesizing **Hybrid Controllers**
- **Variants**
  - Complex dynamics
  - Multi-robot missions
  - High-dimensional systems
- **Specification Languages and Tools for Human Users**
Humans Issuing High Level Orders

- Carry meals to patients
- Deliver medical records
- Patrol patient rooms

http://newsroom.ucla.edu
Humans Issuing High Level Orders

- Carry meals to patients
- Deliver medical records
- Patrol patient rooms

Challenges:
- Easy to instruct
- Does as it is told*

[Image: http://newsroom.ucla.edu]
Structured English to LTL

- **Patrol the aisles.**

  group Corners is r1, r2, r3, r4
  if you are not activating call_manager then visit all Corners
  do look_leftright if and only if
       you are not activating call_manager and
    you were in (between r1 and r2 or between r3 and r4)

- **If you find a missing item, call the manager.**

  call_manager is set on see_missingitem and reset on head_tapped
  if you are activating call_manager then stay there

- **Avoid aisles with spills.**

  spill_top is set on (r1 or r2) and see_spill and reset on false
  spill_bottom is set on (r3 or r4) and see_spill and reset on false

    if you are activating spill_top then always not between r1 and r2
    if you are activating spill_bottom then always not between r3 and r4

"Translating Structured English to Robot Controllers"
[Kress-GazitFP08]
Natural Language

Go to the lounge
Natural Language to LTL

Go to the lounge.

Command: go; Location: lounge

Initially, “lounge” has not been visited.

\[ \neg m_{\text{visit_lounge}} \]

Visit “lounge.”

\[ \square (\bigcirc m_{\text{visit_lounge}} \Leftrightarrow (m_{\text{visit_lounge}} \lor \bigcirc \text{lounge})) \]

“Sorry Dave, I'm Afraid I Can't Do That: Explaining Unachievable Robot Tasks Using Natural Language” [RamanLFMK, RSS'13]  
vasu@caltech.edu  -- ExCAPE Summer School 2015  
63
Explaining Failures

Carry meals from the kitchen to all patient rooms.
Start in the closet. Carry meals from the kitchen to all patient rooms. Don’t go into any public rooms.
Making Life Easier for Human Users

• Natural language pipeline
• Concise cause of unsynthesizability
  – Leveraging SAT minimal unsat cores
  – Explained in natural language
The statements that cause the problem are:
“Carry meals from the kitchen to all patient rooms.” because of item(s): “Deliver ‘meal’ to ‘r1’.”.
“Don’t go into any public rooms.” because of item(s): “Do not go to ‘hall_c’.”.

“Sorry Dave, I'm Afraid I Can't Do That: Explaining Unachievable Robot Tasks Using Natural Language” [RamanLFMK, RSS’13]
Tool!

**LTLMoP**: Linear Temporal Logic Mission Planning Toolkit

[http://ltlmop.github.io/](http://ltlmop.github.io/)
(GPLv3 License)

On Friday

- LTLMoP hands-on exercise
  - [http://ltlmop.herokuapp.com/](http://ltlmop.herokuapp.com/)

- BluSTL demo/hands-on exercise
  - [https://github.com/BluSTL/BluSTL/](https://github.com/BluSTL/BluSTL/)