Automated Software Synthesis for Cyber-Physical Systems

Indranil Saha

UC Berkeley and UPenn
Cyber Physical Systems

Indranil Saha

Automated Software Synthesis for CPS
The systems are mostly life-critical or mission-critical
Embedded Control Software: The Weak Link

Plant

Controller

x' = f(x, u)

u = k(x)

Actuator

Sensor

Control System

Indranil Saha
1962 – Mariner I Space Probe Malfunction
Embedded Control Software: The Weak Link

1962 – Mariner I Space Probe Malfunction

1991 – The Patriot Missile Failure
Embedded Control Software: The Weak Link

- 1962 – Mariner I Space Probe Malfunction
- 1991 – The Patriot Missile Failure
- 1995 – Ariane 5 Flight 501 Explosion

Plant: $x' = f(x, u)$
Controller: $u = k(x)$
Actuator
Sensor
Control System

Indranil Saha
Automated Software Synthesis for CPS
1962 – Mariner I Space Probe Malfunction
1991 – The Patriot Missile Failure
1995 – Ariane 5 Flight 501 Explosion
2014 – Toyota Prius Recall
Today in aerospace industry, control software design, implementation, and testing account for over 60% of the total development cost of an aircraft (Source: Lockheed Martin Aeronautics Company)

1962 – Mariner I Space Probe Malfunction

1991 – The Patriot Missile Failure

1995 – Ariane 5 Flight 501 Explosion

2014 – Toyota Prius Recall

....
Future Cyber-Physical Systems

- **Aerospace**
  - Automatic air-traffic controller

- **Automotive**
  - Self driving car

- **Delivery**
  - Delivery by UAVs

- **HealthCare**
  - Surgical robots

- **Agriculture**
  - Automatic harvestation

- **Entertainment**
  - Robot soccer
Future Cyber-Physical Systems

- **Aerospace**
  - Automatic air-traffic controller

- **Automotive**
  - Self driving car

- **Delivery**
  - Delivery by UAVs

- **HealthCare**
  - Surgical robots

- **Agriculture**
  - Automatic harvestation

- **Entertainment**
  - Robot soccer

More Automation.. More reliance on Software..
Future Cyber-Physical Systems

- **Aerospace**
  - Automatic air-traffic controller
- **Automotive**
  - Self driving car
- **Delivery**
  - Delivery by UAVs
- **HealthCare**
  - Surgical robots
- **Agriculture**
  - Automatic harvestation
- **Entertainment**
  - Robot soccer

More Automation.. More reliance on Software..

**Need of the hour:** More Automation in Software Development
**Research Goal**

To devise tools and technologies to build high-assurance software for cyber-physical systems (CPSs)

**Focus**

Automated Software Synthesis
Example: Multi-Robot Motion Planning

Goal: $I_1 \rightarrow F_1, I_2 \rightarrow F_2,$
$I_3 \rightarrow F_3, I_4 \rightarrow F_4$

Requirements:
- Maintain a rectangular formation
- Maintain a precedence relationship
- Maintain a minimum distance
Planning, Control and Computing Hierarchy

Artificial Intelligence
- Formal Methods

Control Theory

Scheduling Theory
- Software Engineering

High-Level Planning

Control Algorithm Design

Real-Time Software Implementation

- Compositional motion planning for dynamic robots
- Synthesis of Embedded Control Software
Planning, Control and Computing Hierarchy

Artificial Intelligence
- Formal Methods

Control Theory

Scheduling Theory
- Software Engineering

High-Level Planning

Control Algorithm Design

Real-Time Software Implementation

Automated Synthesis with the aid of Formel Methods + Control Theory + Scheduling Theory + Software Engineering
Outline

- Background on Synthesis
- Compositional Motion Planning
- Control Software Synthesis
- Ongoing and Future Work
Outline

- Background on Synthesis
- Compositional Motion Planning
- Control Software Synthesis
- Ongoing and Future Work
Program Synthesis

Specification

System Information

Synthesis Tool
(SMT Solver)

Program

Indranil Saha
Automated Software Synthesis for CPS
Synthesized program is correct-by-construction
Program Synthesis - Specification

Specification

System Information

Synthesis Tool
(SMT Solver)

Program
Should be expressive to capture temporal relationships among the events

**Example:** Visit area $R_2$, then area $R_3$, then area $R_4$, and finally, return and remain in region $R_1$ while avoiding areas $R_2$ and $R_3$

**Linear Temporal Logic**
Linear Temporal Logic (LTL)

LTL Grammar:

\[ \phi ::= \pi | \neg \phi | \phi \land \phi | \diamond \phi | \square \phi | \lozenge \phi | \phi U \phi \]

\( \pi \) - atomic proposition

Example: \( \pi_1 \) - The robot is in Room 1

(next)
\( \diamond \phi \) \( \phi \)

(always)
\( \square \phi \) \( \phi \) \( \phi \) \( \phi \) \( \phi \)

(eventually)
\( \lozenge \phi \) \( \phi \)

(until)
\( \phi_1 U \phi_2 \) \( \phi_1 \) \( \phi_1 \) \( \phi_2 \)
Examples of LTL Specifications

1. Reachability
   \[ \varphi = \lozenge \pi_2 \]

2. Coverage
   \[ \varphi = \lozenge \pi_2 \land \lozenge \pi_3 \land \lozenge \pi_4 \]

3. Sequencing
   \[ \lozenge (\pi_2 \land \lozenge \pi_3) \]

4. Reachability with avoidance
   \[ (\neg \pi_2 \land \neg \pi_3) \cup \pi_4 \]

5. Recurrent sequencing
   \[ \Box \lozenge (\pi_2 \land \lozenge \pi_3) \]

Visit area \( R_2 \), then area \( R_3 \), then area \( R_4 \), and finally, return and remain in region \( R_1 \) while avoiding areas \( R_2 \) and \( R_3 \)

\[ \varphi = \lozenge (\pi_2 \land \lozenge (\pi_3 \land \lozenge (\pi_4 \land (\neg \pi_2 \land \neg \pi_3) \cup \Box \pi_1))))) \]
Program Synthesis - SMT Solver

Specification

System Information

Synthesis Tool

(SMT Solver)

Program

Automated Software Synthesis for CPS
Program Synthesis - SMT Solver

Specification

Synthesis Tool (SMT Solver)

Program

System Information

Z3

GCC

Yices

MathSAT
SMT Solver

**SAT solver:** Checks satisfiability of Boolean formulas

Example: \( b_1 \land b_2 \land (b_2 \rightarrow \neg b_1) \)

**SMT Solver:** SAT solver empowered with Theory solvers

Example Theories: LRA (Linear Real Arithmetic), EUF (Equality with Uninterpreted Functions), ...

Example:
\[
\begin{align*}
x_0 = 0 \land y_0 = 0 \land f(2, 1) = \text{true} \land \\
((x_1 = x_0 + 2) \land (y_1 = y_0 + 1)) \lor ((x_1 = x_0 + 1) \land (y_1 = y_0 + 2)) \land \\
((x_2 = x_1 + 2) \land (y_2 = y_1 + 1)) \lor ((x_2 = x_1 + 1) \land (y_2 = y_1 + 2)) \land \\
f(x_1, y_1) \neq \text{true} \land x_2 \geq 4 \lor y_2 \geq 3
\end{align*}
\]

Formula is satisfiable \( \Rightarrow \) generates a **model**

Formula is unsatisfiable \( \Rightarrow \) generates an **unsatisfiable core**

An SMT solver can be used as an **optimization engine**
- Iteratively search for better solution using binary search
Architecture of a Program Synthesis Tool

- System Information
- Specification

- Constraint Generator
- Constraints
- SMT Solver

- Unsat Core
- Model
- Specification Refinement Tool
- Program Generator

- Program
Outline

- Background on Synthesis
- Compositional Motion Planning
- Control Software Synthesis
- Ongoing and Future Work
Compositional motion planning for dynamic robots

Synthesis of Embedded Control Software
Motion Plan Synthesis

- High-Level Planning
- Control Algorithm Design
- Real-Time Software Implementation

Compositional motion planning for dynamic robots

Synthesis of Embedded Control Software

Specification (LTL formula)
System Information (Dynamics)

Synthesis Tool

Program (Motion Plan)
Existing Solutions for LTL Motion Planning

General Idea:

- Generate a finite abstraction for the robot dynamics
- Generate a finite model for the LTL specification
- Apply a game theoretic algorithm to generate a high level plan
- Generate low level control signals to realize the high level plan

Work by Kress-Gazit, Fainekos, Pappas, Karaman, Frazzoli, Kavraki, Verdi, Topcu, Murray, Belta, Rus and others.

Limitations:

- Discretization of continuous dynamics is expensive
- Existence of low-level controllers is not guaranteed
The goal of ExCAPE is to transform the way programmers develop software by advancing the theory and practice of software synthesis. In the proposed paradigm, a programmer can express insights through a variety of forms such as incomplete programs, example behaviors, and high-level requirements, and the synthesis tool generates the implementation relying on powerful analysis algorithms and programmer collaboration.

The ExCAPE plan is to produce a range of design tools; that let end users program robots by demonstrating example behaviors, and that provide smart assistance for expert programmers to meet challenges in multicore programming.

**ExCAPE Approach to Software Design**

- **Designers**
- **Synthesis Tool**
- **Software**

<table>
<thead>
<tr>
<th>Partial Source Code</th>
<th>Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial Specifications</td>
<td>Libraries</td>
</tr>
<tr>
<td>Platform Constraints</td>
<td>Performance Metrics</td>
</tr>
</tbody>
</table>

Collaboration between:

- Penn University
- Cornell University
- Berkeley University
- Rice University
- UCLA
- University of Maryland
- Massachusetts Institute of Technology
- University of Michigan
- University of Illinois at Urbana-Champaign

Supported by an Expeditions in Computing award from the National Science Foundation.
Motion Primitives

Short, kinematically feasible motions forming the basis of movements of the robot

Components:

- \( u \) - a precomputed control input
- \( \tau \) - the duration for which the control signal is applied
- \( q_i \) - initial velocity configuration
- \( q_f \) - final velocity configuration
- \( X_{rf} \) - relative final position
- \( W \) - the set of relative blocks through which the robot may pass
- \( \text{cost} \) - an estimated energy consumption for executing the control law

Note: Motion Primitives are position oblivious
An input problem instance $\mathcal{P} = \langle R, I, PRIM, Workspace, \xi, L \rangle$

- $R$ - The set of robots
- $I$ - Initial state of the group of robots
- $PRIM = [PRIM_1, PRIM_2, \ldots, PRIM_{|R|}]$
- $Workspace$ - Workspace dimension, position of obstacles
- $\xi$ - Specification given in Linear Temporal Logic
- $L$ - Number of hops in the trajectory

Definition (Motion Planning Problem)
Given an input problem $\mathcal{P}$, synthesize a trajectory of length $L$
Complan
(COMpositional Motion PLANner)
http://www.seas.upenn.edu/~isaha/complan.shtml

$$\Phi(0) \xrightarrow{\text{Prim}_1} \Phi(1) \xrightarrow{\text{Prim}_2} \Phi(2) \ldots \Phi(L - 1) \xrightarrow{\text{Prim}_L} \Phi(L)$$

Constraints: $$(\Phi(0) \in I) \land \|\text{Transition}\| \land \|\text{Specification}\|$$

Boolean combination of constraints from Linear Arithmetic and Equality with Uninterpreted Functions theories

Complan solves for the $L$ motion primitives using an SMT solver
Example: Satisfy Invariants before Reaching Goal

**Goal:**

(I1 and I2) → B
(I3 and I4) → A

**Invariants:**

- Maintain a **rectangular** or **linear** formation
- Maintain a minimum distance
Example: Satisfy Invariants before Reaching Goal

Goal: \((I_1 \text{ and } I_2) \rightarrow B\)  
\((I_3 \text{ and } I_4) \rightarrow A\)

Invariants:
- Maintain a **rectangular or linear** formation
- Maintain a minimum distance

No motion plan that satisfies the formation constraint exists

Unsatisfiable Core helps us refining the specification
Example: Satisfy Invariants before Reaching Goal

Goal: \((I_1 \text{ and } I_2) \rightarrow B \)
\((I_3 \text{ and } I_4) \rightarrow A\)

Invariants:
- Maintain a minimum distance
- The distance between two quadrotors is always greater than one unit
Example: Satisfy Invariants before Reaching Goal

Goal: 
(I1 and I2) → B
(I3 and I4) → A

Invariants:
- Maintain a minimum distance
- The distance between two quadrotors is always greater than one unit

Indranil Saha
Automated Software Synthesis for CPS
Finding Optimal Trajectory

- Find the least number of motion primitives that can generate a valid trajectory

- Among all trajectories that use the least number of motion primitives, find the one that incurs the least cost
Example: Satisfy Invariants before Reaching Goal

Goal: $(I_1 \text{ and } I_2) \rightarrow B$
$(I_3 \text{ and } I_4) \rightarrow A$

Invariants:
- Maintain a minimum distance
- The distance between two quadrotors is always greater than one unit
Example: Satisfy Invariants before Reaching Goal

Goal: (I1 and I2) \(\rightarrow\) B
     (I3 and I4) \(\rightarrow\) A

Invariants:

- Maintain a minimum distance
- The distance between two quadrotors is always greater than one unit
Each quadrotor has to repeatedly gather data from some data gathering location and upload the gathered data at a data upload location.

$$\xi_1 := A(\Box(\Diamond(r_{X\text{gather}} \land (\Diamond r_{X\text{upload}}))))$$
Each quadrotor has to repeatedly gather data from some data gathering location and upload the gathered data at a data upload location.

\[ \xi_1 := A(\Box(\Diamond (rX\text{gather} \land (\Diamond rX\text{upload})))) \]
The TerraSwarm Research Center

The TerraSwarm Research Center, launched on January 15, 2013, is addressing the huge potential (and associated risks) of pervasive integration of smart, networked sensors and actuators into our connected world. The center is funded by the STARnet phase of the Focus Center Research Program (FCRP) administered by the Semiconductor Research Corporation (SRC). Funding comes from the Defense Advanced Research Projects Agency (DARPA) and the SRC industry partners, including GLOBALFOUNDRIES, IBM, Intel Corporation, Micron Technology, Raytheon, Texas Instruments, and United Technologies. (See About the Center, News, Overview Paper, and Research Highlights.)

Upcoming Events

- October 16, 2015: Ptolemy Miniconference (Berkeley).
- October 14-15, 2015: TerraSwarm Annual Meeting (Berkeley).
- April 13, 2015: Second International Workshop on the Swarm at the Edge of the Cloud (Seattle).
- February 12, 2015: TerraSwarm Lunch and Poster Session at the Berkeley EECS Annual Research Symposium, BEARS 2015 (Berkeley).

Recurring Events

- Tuesdays, 12noon-1pm (Pacific): We hold theme-specific teleconferences on the first four Tuesdays of the month.
- Thursdays, 12:30-2 (Pacific): Seminar Series, 490 Cory Hall, co-sponsored with the Ubiquitous SwarmLab.

News and Events

- Ongoing: News items about TerraSwarm Researchers from the media as reported in the TerraSwarm Blog.
- January 16 & 17, 2015: Test Bed Workshop (Berkeley).
- October 29 & 30, 2014: TerraSwarm Annual Meeting (Berkeley)
Terraswarm Themes

Task D.1.1: TerraSwarm Infrastructure
Task D.1.2: TerraSwarm Applications

Theme 1: Smart City

Task D.2.1: The SwarmOS
Task D.2.2: Resource Optimization with Dynamic and Uncertain Access
Task D.2.3: Networks of Services
Task D.2.4: Sensor and Actuator Component Architecture
Task D.2.5: Energy Efficient Roots of Trust for Sensors and Actuators

The TerraSwarm Research Center (TSRC)

Task D.4.1: Design Methodologies: Contracts and Mapping
Task D.4.2: Programming Models and Modeling Formalisms
Task D.4.3: Model Construction, Synthesis and Learning
Task D.4.4: Verification and Security Analysis

Theme 4: Methodologies, Models, and Tools

Theme 2: Platform Architectures and Operating Systems

Theme 3: Services and Cloud Interaction
Task D.3.1: Data to Information
Task D.3.2: Localization in TerraSwarm Systems
Task D.3.3: Control of Mobile Vehicles and Other Dynamic Assets
Task D.3.4: Secure and Privacy-Preserving Monitoring and Control in the Cloud
Specification: \( \neg obstacles \cup reach \)
Motion Plan Synthesis for a Swarm of Robots

Specification: $\neg \text{obstacles} \cup \text{reach}$

Main Idea:
- Synthesize optimal trajectory for each robot independently
Specification: $\neg obstacles \cup reach$

Main Idea:
- Synthesize optimal trajectory for each robot independently
Motion Plan Synthesis for a Swarm of Robots

Specification: $\neg obstacles \cup reach$

Main Idea:
- Synthesize optimal trajectory for each robot independently
- Find a feasible ordering for the robots
Motion Plan Synthesis for a Swarm of Robots

Specification: $\neg obstacles \cup reach$

Main Idea:
- Synthesize optimal trajectory for each robot independently
- Find a feasible ordering for the robots
- Synthesize final trajectories according to the assigned priorities
  - Treat the robots with higher priorities as dynamic obstacles
  - Introduce minimum delay to execute the optional trajectory to avoid collision
Motion Plan Synthesis for a Swarm of Robots

Specification: \( \neg obstacles \cup reach \)

**Implan**
(Incremental Motion PLANner)

**Main Idea:**
- Synthesize optimal trajectory for each robot independently
- Find a feasible ordering for the robots
- Synthesize final trajectories according to the assigned priorities
  - Treat the robots with higher priorities as dynamic obstacles
  - Introduce minimum delay to execute the optional trajectory to avoid collision
Motion Plan Synthesis for a Swarm of Robots

Ordering:
1: $R_3$  2: $R_6$  3: $R_2$
4: $R_4$  5: $R_1$  6: $R_5$
Motion Plan Synthesis for a Swarm of Robots

<table>
<thead>
<tr>
<th></th>
<th>i1</th>
<th>i2</th>
</tr>
</thead>
<tbody>
<tr>
<td>i3</td>
<td>f1</td>
<td>f4</td>
</tr>
<tr>
<td>i4</td>
<td>f5</td>
<td>f6</td>
</tr>
<tr>
<td>i5</td>
<td>i6</td>
<td></td>
</tr>
</tbody>
</table>

Ordering:
1: $R_3$  2: $R_6$  3: $R_2$
4: $R_4$  5: $R_1$  6: $R_5$

Challenge:
What if an ordering does not exist?
Motion Plan Synthesis for a Swarm of Robots

Ordering:
1: $R_3$    2: $R_6$    3: $R_2$
4: $R_4$    5: $R_1$    6: $R_5$

Challenge:
What if an ordering does not exist?
25 quadrotors moving in a closed place

Specification:
\( \neg \text{obstacles} \cup \text{reach} \)
Outline

- Background on Synthesis
- Compositional Motion Planning
- Control Software Synthesis
- Ongoing and Future Work
Control Software Synthesis

- High-Level Planning
- Control Algorithm Design
- Real-Time Software Implementation

Compositional motion planning for dynamic robots

Synthesis of Embedded Control Software
Control Software Synthesis

High-Level Planning

Control Algorithm Design

Real-Time Software Implementation

Compositional motion planning for dynamic robots

Synthesis of Embedded Control Software

Specification (Stability)

System Information (Model of the Plant)

Platform Information (Number of Bits)

Synthesis Tool

Program (Controller Software)
The plant converges to a desired behavior under the actions of the controller.
The plant converges to a desired behavior under the actions of the controller.

Example:
In the steady state, the quadrotor will be at 4m away from the x-axis and 6m away from the y-axis.
Specification for Implemented Controller: Practical Stability

The state of the plant eventually reaches a bounded region and remains there under the action of the controller.

Specification is given in terms of Region of Practical Stability.
Mathematical Model

Software Implementation

The state of the plant eventually reaches a bounded region and remains there under the action of the controller.

Specification is given in terms of Region of Practical Stability.

Example:
In the steady state, the quadrotor will be between 3.9-4.1m away from the x-axis and 5.8-6.2m away from the y-axis.
Given:
- Dynamics of the plant
- A stabilizing controller
- The number of bits of the target processor
- Specification on practical stability

Verify: If the software implementation of the controller satisfies the specification on practical stability

Reduces to computing the upper bound on the region of practical stability
Bound on the Region of Practical Stability

Mathematical Model

Software Implementation
Boundary on the Region of Practical Stability

Mathematical Model

Software Implementation

**Theorem [EMSOFT2010]** If $\gamma$ is the L2-Gain of a control system, $b$ is a bound on the implementation error, then

$$\rho \leq \gamma \times b$$
**Theorem [EMSOFT2010]** If $\gamma$ is the L2-Gain of a control system, $b$ is a bound on the implementation error, then

$$\rho \leq \gamma \times b$$

**Separation of concerns:**

- Compute L2-gain from the mathematical model (standard problem in control theory)
- Compute the bound on implementation error (analysis of the implementation)
Example of Controller Program

Control Law (Vehicle Steering):

\[ u = 0.81 \times (\text{In1} - \text{In2}) - 1.017 \times \text{In3} \]

Real-valued program

```c
Real In1, In2, In3;
Real Subtract, Gain, Gain2, Out1;

static void output(void) {
    Subtract = In1 - In2;
    Gain = 0.81 * Subtract;
    Gain2 = 1.017 * In3;
    Out1 = Gain - Gain2;
}
```

Fixed-point implementation (16-bit):

```c
short int In1, In2, In3;
short int Subtract, Gain, Gain2, Out1;

static void output(void) {
    Subtract = (short int)(In1 - In2);
    Gain = (short int)(26542 * Subtract >> 15);
    Gain2 = (short int)(16663 * In3 >> 14);
    Out1 = (short int)(((Gain << 1) - Gain2) >> 1);
}
```
Example of Controller Program

Control Law (Vehicle Steering):
\[ u = 0.81 \times (\text{In1} - \text{In2}) - 1.017 \times \text{In3} \]

### Real-valued program

Real In1, In2, In3;
Real Subtract, Gain, Gain2, Out1;

static void output(void) {
    Subtract = In1 - In2;
    Gain = 0.81 \times \text{Subtract};
    Gain2 = 1.017 \times \text{In3};
    Out1 = \text{Gain} - \text{Gain2};
}

### Fixed-point implementation (16-bit):

short int In1, In2, In3;
short int Subtract, Gain, Gain2, Out1;

static void output(void) {
    Subtract = (\text{short int})(\text{In1} - \text{In2});
    Gain = (\text{short int})(26542 \times \text{Subtract} \gg 15);
    Gain2 = (\text{short int})(16663 \times \text{In3} \gg 14);
    Out1 = (\text{short int})(((\text{Gain} \ll 1) - \text{Gain2}) \gg 1);
}

What is the bound on the error?

Can be formulated as an SMT solving problem over Boolean combination of linear arithmetic constraints.
Controller Stability Analyzer: Costan

- An automatic tool to compute the bound on the region of practical stability

- Supports both linear and nonlinear controllers, for nonlinear controllers both polynomial implementation and lookup table based implementation
Is it possible to synthesize a controller that minimizes the region of practical stability?
Is it possible to synthesize a controller that minimizes the region of practical stability?

Traditionally, controllers are synthesized minimizing LQR cost function

LQR cost = state cost + control cost
state cost = sum of the deviations of the states from their desired values
control cost = energy expended by the control action
Model of a Vehicle Steering:

\[
\begin{bmatrix}
\dot{\xi}_1 \\
\dot{\xi}_2
\end{bmatrix} =
\begin{bmatrix}
0 & \frac{g}{h} \\
1 & 0
\end{bmatrix}
\begin{bmatrix}
\xi_1 \\
\xi_2
\end{bmatrix} +
\begin{bmatrix}
1 \\
0
\end{bmatrix} (\nu + \omega)
\]

\[
y = \begin{bmatrix}
\frac{av_0}{bh} & \frac{v_0^2}{bh}
\end{bmatrix}
\begin{bmatrix}
\dot{\xi}_1 \\
\dot{\xi}_2
\end{bmatrix} + \nu
\]

LQR Controller:

\[K_1 = [5.1538, 12.9724]\]

LQR cost function is **264.1908**

Another Controller:

\[K_2 = [3.0253, 12.6089]\]

LQR cost function is **284.1578**
There is a trade-off between LQR cost and the region of practical stability
Problem Statement

Design a controller optimizing the following objectives:

- The LQR cost
- The bound on the region of practical stability
Controller Synthesis Tool: Ocsyn

- Co-optimizes LQR cost and the bound on region of practical stability
- Employs stochastic optimization — needs to compute the value of the objective functions for many different controllers
  - Employs a convex optimization tool to compute LQR cost
  - Uses Costan to compute the region of practical stability
- Synthesizes controller
  - with significant improvement on the bound on region of practical stability
  - without significant degradation on the LQR cost

**Example:** vehicle steering
A factor of 10 improvement in the region of practical stability with only 10% degradation in LQR cost
Outline

- Background on Synthesis
- Compositional Motion Planning
- Control Software Synthesis
- Ongoing and Future Work
Ongoing Research

Complan:

- New class of properties
  - Timing specification
- New System Model
  - Mixed Logical Dynamical System
  - Time varying motion primitives

Implan:

- Linear Temporal Logic Motion Planning for large number of robots
**Goal:** Devise tools and techniques for synthesizing motion planner that is

- **Reactive**
  Self-Driven Car, Robocup

- **Robust**
  Path planning for a drone in the presence of gust of wind

- **Distributed**
  Path planning for an aircraft in a congested airspace

**Challenges:**

- How to make existing solvers more amenable to solve planning problems?

- How to build domain specific solvers? (e.g., a solver that has control theoretic intelligence)
Application of Big Data analytics in cyber-physical systems
  - Automatic air-traffic controller
  - Real-time decision making by unmanned vehicles

Security for cyber-physical systems
  - Devise defense against attacks on unmanned aerial vehicles
Thank You!!

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