Automated Software Synthesis for Complex Robotic Systems

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Complex Robotic Systems
The systems are mostly life-critical or mission-critical
Embedded Control Software: The Weak Link

Plant
\( \dot{x} = f(x, u) \)

Controller
\( u = k(x) \)

Actuator
Sensor
Control System

Indranil Saha
Embedded Control Software: The Weak Link

Plant

\[ \dot{x} = f(x, u) \]

Controller

\[ u = k(x) \]

Actuator

Sensor

Control System

Plant

Controller

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

Plant

x’ = f(x, u)

Controller

u = k(x)

Actuator Sensor

Control System

1962 – Mariner I Space Probe Malfunction

1991 – The Patriot Missile Failure
1962 – Mariner I Space Probe Malfunction

1991 – The Patriot Missile Failure

1995 – Ariane 5 Flight 501 Explosion
Embedded Control Software: The Weak Link

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 Robotic Systems

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

- **Automotive**
  - Self driving car

- **Delivery**
  - Delivery by UAVs

- **HealthCare**
  - Surgery by robot arms

- **Agriculture**
  - Automatic crop and fruit harvestating

- **Entertainment**
  - Robot soccer, robot music band
Future Robotic Systems

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

- **Automotive**
  - Self driving car

- **Delivery**
  - Delivery by UAVs

- **HealthCare**
  - Surgery by robot arms

- **Agriculture**
  - Automatic crop and fruit harvestating

- **Entertainment**
  - Robot soccer, robot music band

More Automation.. More reliance on Software..
Future Robotic Systems

- Aerospace
  - Automatic air-traffic controller
- Automotive
  - Self driving car
- Delivery
  - Delivery by UAVs
- HealthCare
  - Surgery by robot arms
- Agriculture
  - Automatic crop and fruit harvestating
- Entertainment
  - Robot soccer, robot music band

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 complex robotic systems

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

High-Level Planning

Control Theory

Control Algorithm Design

Scheduling Theory
Software Engineering

Real-Time Software Implementation

Compositional motion planning for dynamic robots

Synthesis of Embedded Control Software
Planning, Control and Computing Hierarchy

Artificial Intelligence
Formal Methods

High-Level Planning

Compositional motion planning for dynamic robots

Control Theory

Control Algorithm Design

Synthesis of Embedded Control Software

Scheduling Theory
Software Engineering

Real-Time Software Implementation

Research Focus

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

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

Synthesis Tool (SMT Solver)

- Specification
- System Information

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 (LTL)

LTL Grammar:

\[ \phi ::= \pi | \neg \phi | \phi \land \phi | \Diamond \phi | \Box \phi | \phi U \phi \]

- \( \pi \) - atomic proposition

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

(next)

\[ \Box \phi \]

(always)

\[ \Box \phi \]

(eventually)

\[ \Diamond \phi \]

(until)

\[ \phi_1 U \phi_2 \]
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

Synthesis Tool (SMT Solver)

Specification
System Information
Program
Program Synthesis - SMT Solver

Specification

System Information

Synthesis Tool
(SMT Solver)

Program

Z3

Yices

GCC

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

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

1. Constraint Generator
2. Constraints
3. SMT Solver
4. Unsat Core
5. Model
6. Specification Refinement Tool
7. Program Generator
8. Program
Outline

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

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

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)
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
Motion Planning Problem

An input problem instance \( P = \langle R, I, PRIM, Workspace, \xi, L \rangle \)

- **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 \( P \), synthesize a trajectory of length \( L \)
Complan

(COMpositional Motion PLANner)
http://www.cse.iitk.ac.in/~isaha/complan.shtml

Φ(0) →^\text{Prim}_1 Φ(1) →^\text{Prim}_2 Φ(2) \ldots Φ(L - 1) →^\text{Prim}_L Φ(L)

Constraints: (Φ(0) ∈ I) ∧ ⌊Transition⌋ ∧ ⌊Specification⌋

Boolean combination of constraints from \text{Linear Arithmetic} and \text{Equality with Uninterpreted Functions} theories

Complan solves for the \( L \) motion primitives using an \text{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
**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: (I1 and I2) → B
    (I3 and I4) → A

Invariants:

- Maintain a minimum distance
- The distance between two quadrotors is always greater than one unit
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
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: (I1 and I2) → B
     (I3 and I4) → 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
Each quadrotor has to repeatedly gather data from some data gathering location and upload the gathered data at a data upload location.

\[ \xi_1 := \mathcal{A}(\Box(\Diamond(rX_{gather} \land (\Diamond rX_{upload})))) \]
Each quadrotor has to repeatedly gather data from some data gathering location and upload the gathered data at a data upload location.

\[ \xi_1 := A(\Box(rX_{gather} \land (\Diamond rX_{upload}))) \]
Specification: $\neg \text{obstacles} \cup \text{reach}$
Specification: $\neg \text{obstacles} \cup \text{reach}$

Main Idea:
- Synthesize optimal trajectory for each robot independently
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
- Find a feasible ordering for the robots
Specification: \( \neg \text{obstacles } \cup \text{ reach} \)

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

**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

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?
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?
Example: Motion Planning in a Compact Workspace

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

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.
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

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) :
\[ u = 0.81 \times (I_{n1} - I_{n2}) - 1.017 \times I_{n3} \]

Real-valued program

Real \( I_{n1}, I_{n2}, I_{n3}; \)
Real Subtract, Gain, Gain2, Out1;

static void output(void) {
    Subtract = I_{n1} - I_{n2};
    Gain = 0.81 \times \text{Subtract};
    Gain2 = 1.017 \times I_{n3};
    Out1 = \text{Gain} - \text{Gain2};
}

Fixed-point implementation (16-bit):

short int \( I_{n1}, I_{n2}, I_{n3}; \)
short int Subtract, Gain, Gain2, Out1;

static void output(void) {
    Subtract = (\text{short int})(I_{n1} - I_{n2});
    Gain = (\text{short int})(26542 \times \text{Subtract} \gg 15);
    Gain2 = (\text{short int})(16663 \times I_{n3} \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
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

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

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

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

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.
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.cse.iitk.ac.in/~isaha