Principles of Robot Autonomy II

Specifications, Model Checking, and Reactive Synthesis
Today’s itinerary

• Safety from the lens of formal methods

• Model checking & Reactive Synthesis

• Linear Temporal Logic

• Reactive synthesis from Linear Temporal Logic

• Limitations
Today’s itinerary

• Safety from the lens of formal methods

• Model checking & Reactive Synthesis

• Linear Temporal Logic

• Reactive synthesis from Linear Temporal Logic

• Limitations
What does safety mean for Robotics?
Mathematically based technique for specification, development, and verification of software and hardware systems.

**Verification:** Timing and Energy analysis

**Software:** Embedded code

Timing Analysis of Interrupt-Driven Programs under Context Bounds
Jonathan Kotker, Dorsa Sadigh, Sanjit A. Seshia
Mathematically based technique for specification, development, and verification of software and hardware systems.

Verification (model checking):

- Specification
- Model
- Verifier

Model satisfies specification? Yes / No
What does safety mean for robots?

Formal methods (verification) + Control Theory, Robotics, Machine Learning

Can we prove that autonomous cars are safe?
Formal Methods

Mathematically based technique for specification, development, and verification of software and hardware systems.

Verification (model checking):

- Specification
- Model
- Verifier

Yes / No
Church’s Problem

- Alonzo Church at the “Summer Institute of Symbolic Logic”, Cornell University, 1957:

“Given a requirement which a circuit is to satisfy, we may suppose the requirement expressed in some suitable logistic system which is an extension of restricted recursive arithmetic. The synthesis problem is then to find recursion equivalences representing a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit).”
Formal Methods

Mathematically based technique for specification, development, and verification of software and hardware systems.

Reactive Synthesis (Church 1957):
Formal Methods

**Reactive Synthesis (Church 1957):**

Mathematically based technique for specification, development, and verification of software and hardware systems.
Can we synthesize controller for autonomous cars are provably safe?

Formal methods (verification) + Control Theory, Robotics, Machine Learning

What does safety mean for robots?
Reactive Synthesis

• Automatically constructing correct systems from specifications from scratch.

**Specification:** Temporal behavior specified over input and output sequences

**Reactive System:**
- Every step it reads inputs and produces outputs synchronously
- Non-terminating behavior
Reactive Synthesis

Determine

• If there is a finite state system that realizes the specification (realizability).

• If a specification is realizable, how do you construct the system?
Reactive Synthesis

Church 1957:
• Introduced the reactive synthesis problem

Buchi + Landweber 1969:
• Realizability is decidable.
• If a winning strategy exists, then a finite state winning strategy exists.
• Introduced a game view of the problem.

Rabin 1972:
• Simpler solution to the game with Rabin Tree Automaton.

Pnueli + Rosner 1989:
• Linear Temporal Logic Synthesis
• GR1 Synthesis (Piterman, Pnueli, Sa’ar 2006)
Today’s itinerary

• Safety from the lens of formal methods
• Model checking & Reactive Synthesis
• Linear Temporal Logic
• Reactive synthesis from Linear Temporal Logic
• Limitations
Linear Temporal Logic

\[ \varphi ::= true \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid X\varphi \mid \varphi_1 U \varphi_2 \]

\( a \): atomic proposition

\( X \): \( \varphi \) is true at next step.

\( U \): \( \varphi_1 \) is true until \( \varphi_2 \) at some point becomes true.
Additional Operators

\[ F \varphi := \text{true} U \varphi \] \qquad \varphi \text{ will become true at some point in the future.}

\[ G \varphi := \neg F \neg \varphi \] \qquad \varphi \text{ is always true.}

Eventually/Future: \( F a \)

Always/Globally: \( G a \)

Other combinations:

- \( GF a \): infinitely often
- \( FG a \): notion of stability
- \( G(a \rightarrow F b) \): response
Linear Temporal Logic

‘Temporal’ refers to the underlying nature of time.

- **Linear temporal logic (LTL)**: each moment in time has a well-defined successor moment.
- **Branching temporal logic**: reason about multiple possible time courses (parallelism), e.g., computation tree logic (CTL).
Today’s itinerary

• Safety from the lens of formal methods
• Model checking & Reactive Synthesis
• Linear Temporal Logic
• Reactive synthesis from Linear Temporal Logic
• Limitations
Reactive Synthesis from Linear Temporal Logic

$I$ : input variables, $O$ : output variables

**Game:**
System chooses from $2^O$.
Environment chooses from $2^I$.

**Infinite Play:**
$i_0, i_1, i_2, ...$
$o_0, o_1, o_2, ...$

**Infinite Behavior:** $i_0 \cup o_0, i_1 \cup o_1, i_2 \cup o_2, ...$

**Win:** Behavior $\models$ spec

**Specifications:** LTL formula on $I \cup O$

**Strategy:** function $f : (2^I)^* \rightarrow 2^O$
The Coffee Machine Problem:

“If the start button is pressed, then the system will immediately start grinding for the next two cycles and, then stops grinding and starts brewing for the next two cycles after that, coffee will be produced.”

The specification is unrealizable, because the environment can produce input that makes it impossible to satisfy both requirements at the same time.
Synthesizing Systems

1. Specification
   Logical formulas + Partitioning

2. Construct Game

3. Solve Game
   Finding Controller

4. Correct System

System with Freedom
Plant
Word automaton

\( G(r \rightarrow Xg) \)

Game

Strategy / Mealy machine