Synthesis for Multi-Robot Controllers with Interleaved Motion

Vasu Raman\textsuperscript{1}
Hadas Kress-Gazit\textsuperscript{2}

\textsuperscript{1}California Institute of Technology
\textsuperscript{2}Cornell University

ICRA
3 June 2014
Multi-Robot High-Level Tasks

- Teams that operate **autonomously**
- Fulfill **complex** requirements
- Easy to **specify** and **enforce** guarantees

Vasu Raman  
(vasu@caltech.edu)
Formal Methods: Two Perspectives

Verification

System Model $\mathcal{M}$

Specification $\varphi$

Proof Engine

Proof whether $\mathcal{M} \models \varphi$

“satisfies”

Synthesis

Specification $\varphi$

Synthesis Engine

System $\mathcal{M}$ such that $\mathcal{M} \models \varphi$

Vasu Raman
(vasu@caltech.edu)
Formal Methods: Two Perspectives

Verification

System Model $\mathcal{M}$

Specification $\varphi$

Proof Engine

Proof whether $\mathcal{M} \models \varphi$

Synthesis

Specification $\varphi$

Synthesis Engine

System $\mathcal{M}$ such that $\mathcal{M} \models \varphi$

Model Checkers, Theorem Provers, etc.
Formal Methods: Two Perspectives

Verification

- System Model $\mathcal{M}$
- Specification $\varphi$
- Proof Engine
- Proof whether $\mathcal{M} \models \varphi$
- Model Checkers, Theorem Provers, etc.

Synthesis

- Specification $\varphi$
- Synthesis Engine
- System $\mathcal{M}$ such that $\mathcal{M} \models \varphi$
Reactive Synthesis

\[
\text{Specification } \varphi \\
\text{Synthesis Engine} \\
\text{System } \mathcal{M} \text{ such that } \mathcal{M} \models \varphi
\]
Reactive Synthesis

- **Specification**
  \[ \varphi = \varphi_e \implies \varphi_s \]

- **Synthesis Engine**

- **System** \( M \) such that \( M \models \varphi \)
Reactive Synthesis

• If the operating environment obeys $\varphi_e$, the system satisfies $\varphi_s$. 

![Diagram]

Specification

$\varphi = \varphi_e \Rightarrow \varphi_s$

Synthesis Engine

System $\mathcal{M}$ such that $\mathcal{M} \models \varphi$

Vasu Raman
(vasu@caltech.edu)
Reactive Synthesis

- If the operating environment obeys $\varphi_e$, the system satisfies $\varphi_s$. 

\[ \varphi = \varphi_e \Rightarrow \varphi_s \]

Multi-robot systems

Vasu Raman  
(vasu@caltech.edu)
Reactive Synthesis

• If the operating environment obeys $\varphi_e$, the system satisfies $\varphi_s$.

• Today: specifications in Linear Temporal Logic (LTL)*
  - Propositional logic +
    - $\mathcal{U}$ (until) $\bigcirc$ (next)
    - $\square$ (always) $\diamond$ (eventually)

*actually, GR(1)
LTL Synthesis for Multi-Robot Systems (Related Work)

- Kloetzer and Belta, T-Ro 2007, 2010
  - not reactive

- Chen & Belta, T-Ro 2010
  - smaller class of specifications

- Loizou and Kyriakopoulos, CDC 2004
  - restrict non-motion actions to be continuous

- Kress-Gazit, Ayanian, Pappas & Kumar, CASE 2008
  - restrict robot motion to one at a time

Vasu Raman
(vasu@caltech.edu)
LTL Synthesis for Robot Control

Create abstraction of problem

Synthesize correct discrete solution

Continuously implement discrete solution
**LTL Synthesis for Robot Control**

1. **Create abstraction of problem**
   - Boolean propositions representing sensing & actuation events + LTL specification

2. **Synthesize correct discrete solution**
   - Finite State Machine such that $\mathcal{M} \models \varphi$

3. **Continuously implement discrete solution**
   - Hybrid controller executing low-level actions according to FSM

Vasu Raman
(vasu@caltech.edu)
Example: Recycling Robots

- Robot 1 starts in r1, robot 2 in r4. Neither is carrying anything.

- Robot 1 activates carrying_glass if sensing glass, carrying_metal if sensing metal.

- If Robot 1 is carrying_glass, it should visit r4 and if carrying_metal, it should visit r2.

- Robot 1 always eventually goes to r4. Robot 2 always eventually goes to r1.
Example: Recycling Robots

- Robot 1 starts in r1, robot 2 in r4. Neither is carrying anything.

- Robotic 1 activates carrying_glass if sensing glass, carrying_metal if sensing metal.

- If Robot 1 is carrying_glass, it should visit r4 and if carrying_metal, it should visit r2.

- Robot 1 always eventually goes to r4. Robot 2 always eventually goes to r1.
Example: Recycling Robots

- Robot 1 starts in r1, robot 2 in r4. Neither is carrying anything.

- Robot 1 activates \texttt{carrying\_glass} if sensing glass, \texttt{carrying\_metal} if sensing metal.

- If Robot 1 is \texttt{carrying\_glass}, it should visit r4 and if \texttt{carrying\_metal}, it should visit r2.

- Robot 1 always eventually goes to r4. Robot 2 always eventually goes to r1.

For each robot $i$:

**Actions:**
- Motion to region r1-4 $\pi_{i,r_j}$
- $\pi_{i,\text{carrying\_glass}}$, $\pi_{i,\text{carrying\_metal}}$

**Sensors:**
- glass $\pi_{i,\text{glass}}$
- metal $\pi_{i,\text{metal}}$
Example: Recycling Robots

• Robot 1 starts in r1, robot 2 in r4. Neither is carrying anything.

\[
(\pi_{1r_1} \land \pi_{2r_4} \\
\land \neg \pi_{1\text{carrying\_metal}} \land \neg \pi_{1\text{carrying\_glass}} \\
\land (\neg \pi_{2\text{carrying\_metal}} \land \neg \pi_{2\text{carrying\_glass}})
\]

• Robot 1 activates carrying\_glass if sensing glass, carrying\_metal if sensing metal.

\[
(\Box (\Diamond \pi_{1\text{glass}} \Rightarrow \Diamond \pi_{1\text{carrying\_glass}}) \\
\land (\Box (\Diamond \pi_{1\text{metal}} \Rightarrow \Diamond \pi_{1\text{carrying\_metal}})
\]

• If Robot 1 is carrying\_glass, it should visit r4 and if carrying\_metal, it should visit r2.

\[
(\Box \Diamond (\pi_{1\text{carrying\_glass}} \Rightarrow \Diamond \pi_{1r_4}) \\
\land (\Box \Diamond (\pi_{1\text{carrying\_metal}} \Rightarrow \Diamond \pi_{1r_2})
\]

• Robot 1 always eventually goes to r4. Robot 2 always eventually goes to r1.

\[
(\Box \Diamond (\pi_{1r_4}) \land (\Box \Diamond (\pi_{2r_1})
\]

Vasu Raman
(vasu@caltech.edu)
Motion Control

• Atomic motion controllers [Ayanian & Kumar, T-Ro 2010]
  – Polytope graph on team configuration space
    • (includes proximity constraints)

[Kress-Gazit, Ayanian, Pappas & Kumar, CASE 2008]
Motion Control

• Atomic motion controllers [Ayanian & Kumar, T-Ro 2010]
  – Polytope graph on team configuration space
    • (includes proximity constraints)

• Bisimulation Condition
  – controller exists for every admissible discrete transition

[Kress-Gazit, Ayanian, Pappas & Kumar, CASE 2008]
Motion Control

• Atomic motion controllers [Ayanian & Kumar, T-Ro 2010]
  – Polytope graph on team configuration space
    • (includes proximity constraints)

• Bisimulation Condition
  – controller exists for every admissible discrete transition

Only one robot crosses a room threshold at a time
Naïve Approach

• Restrict discrete transition system
  – If robot 1 is moving, robot 2 stays in place

Spec summary:
• Robot 1 starts in r1, robot 2 in r4.
• Robot 1 does a bunch of recycling tasks and also always eventually goes to r4.
• Robot 2 always eventually goes to r1.
Naïve Approach

• Restrict discrete transition system
  – If robot 1 is moving, robot 2 stays in place

Spec summary:
• Robot 1 starts in r1, robot 2 in r4.
• Robot 1 does a bunch of recycling tasks and also always eventually goes to r4.
• Robot 2 always eventually goes to r1.

• Too conservative
  – If robot 1 (worker) is always moving, robot 2 (freeloader) can never reach r1 \(\rightarrow\) synthesis fails

Vasu Raman
(vasu@caltech.edu)
Solution Overview

Create abstraction of problem → Synthesize correct discrete solution → Continuously implement discrete solution

- Explicitly model motion initiation/completion

**Actions:** \( \pi_{i} \land \pi_{j} \land \pi_{i_{\text{carrying\_glass}}} \land \pi_{i_{\text{carrying\_metal}}} \)

**Sensors:** \( \pi_{i_{\text{glass}}} \land \pi_{i_{\text{metal}}} \land \pi_{i_{\text{carrying\_glass}}} \land \pi_{i_{\text{carrying\_metal}}} \)

"Robot 1 starts in r1, robot 2 in r4."

\[ \pi_{1_{r1}}^C \land \pi_{2_{r4}}^C \] (instead of \( \pi_{1_{r1}} \land \pi_{2_{r4}} \))

"Robot 1 always eventually goes to r4."

\[ \blacklozenge \quad \square \quad \pi_{1_{r4}}^C \] (instead of \( \blacklozenge \quad \square \quad \pi_{1_{r4}} \))
Solution Overview

Enhanced GR(1) Synthesis Algorithm

[Raman, Piterman and Kress-Gazit, ICRA 13]

• Accommodates “transition” goals
  \[ \square \Diamond (\varphi_1 \Rightarrow \lozenge \varphi_2) \] in addition to \[ \square \Diamond \varphi \]

• Efficient for new specifications
Solution Overview

- Simultaneous motion \textit{initiation} for all robots
- At most one robot \textit{completes} region change at a time
Example (finite state machine)

Robot 1 in r3, moving towards r1.

Robot 2 in r2, moving towards r4.

Robot 1 arrives in r1.

Robot 2 arrives in r4.

Both robots stay put.
Example (simulation)

Vasu Raman
(vasu@caltech.edu)
Next Steps

• Experimental validation on hardware

• More concise discrete abstraction
  – Currently adding one sensor proposition per action
Thanks!

Synthesis for Multi-Robot Controllers with Interleaved Motion

Vasu Raman\textsuperscript{1}
Hadas Kress-Gazit\textsuperscript{2}

\textsuperscript{1}California Institute of Technology
\textsuperscript{2}Cornell University

Contact: vasu@caltech.edu

ICRA
3 June 2014