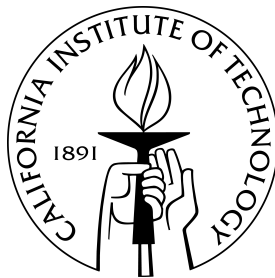


Towards Minimal Explanations of Unsynthesizability for High-Level Robot Behaviors

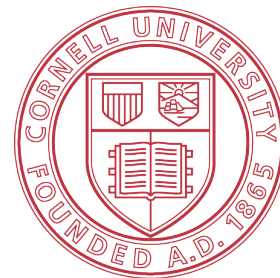
Vasu Raman

Department of Computing and
Mathematical Sciences
California Institute of Technology



Hadas Kress-Gazit

Sibley School of Mechanical and
Aerospace Engineering
Cornell University





High-Level Tasks:
Carrying meals to patients
Delivering medical records
Patrolling patient rooms

.
. .
. .

http://newsroom.ucla.edu/portal/ucla/artwork/4/5/7/9/4/245794/EVA_Robot_3-c.jpg



High-Level Tasks:

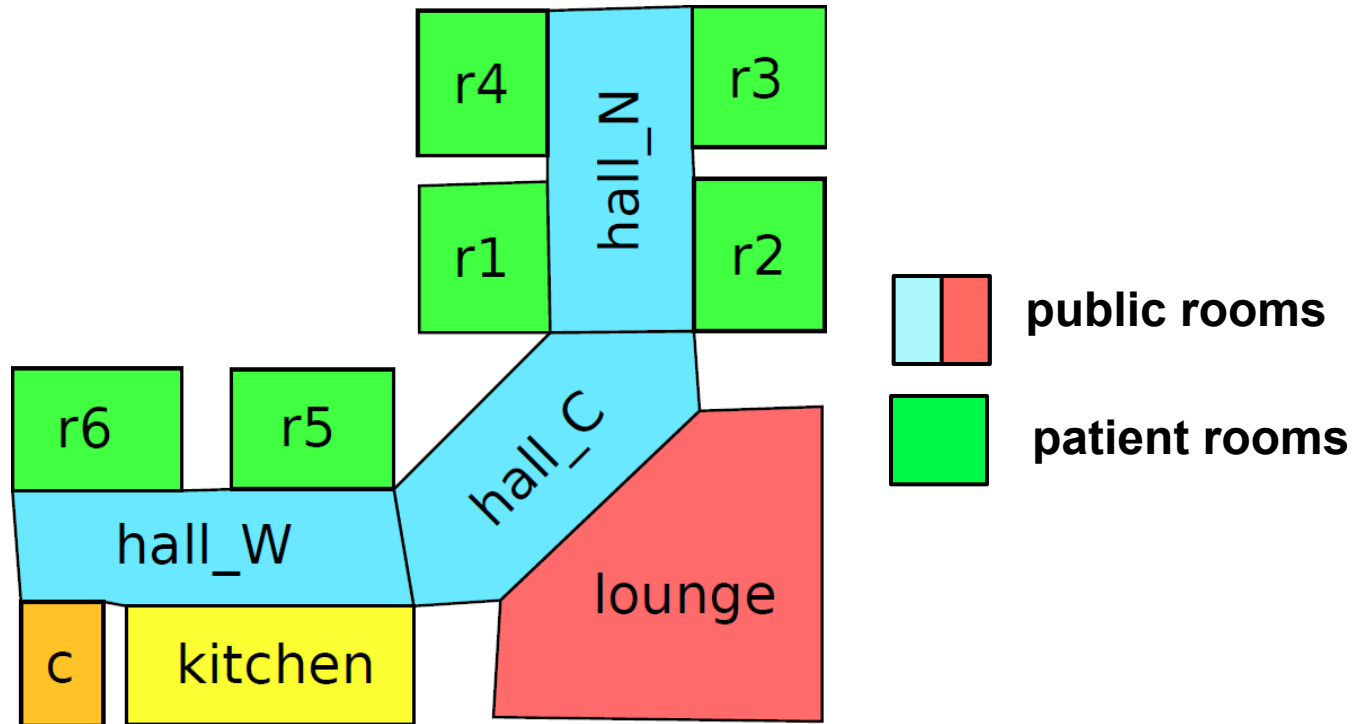
Carrying meals to patients
Delivering medical records
Patrolling patient rooms

Challenges:

Easy to instruct
Does as it is told*

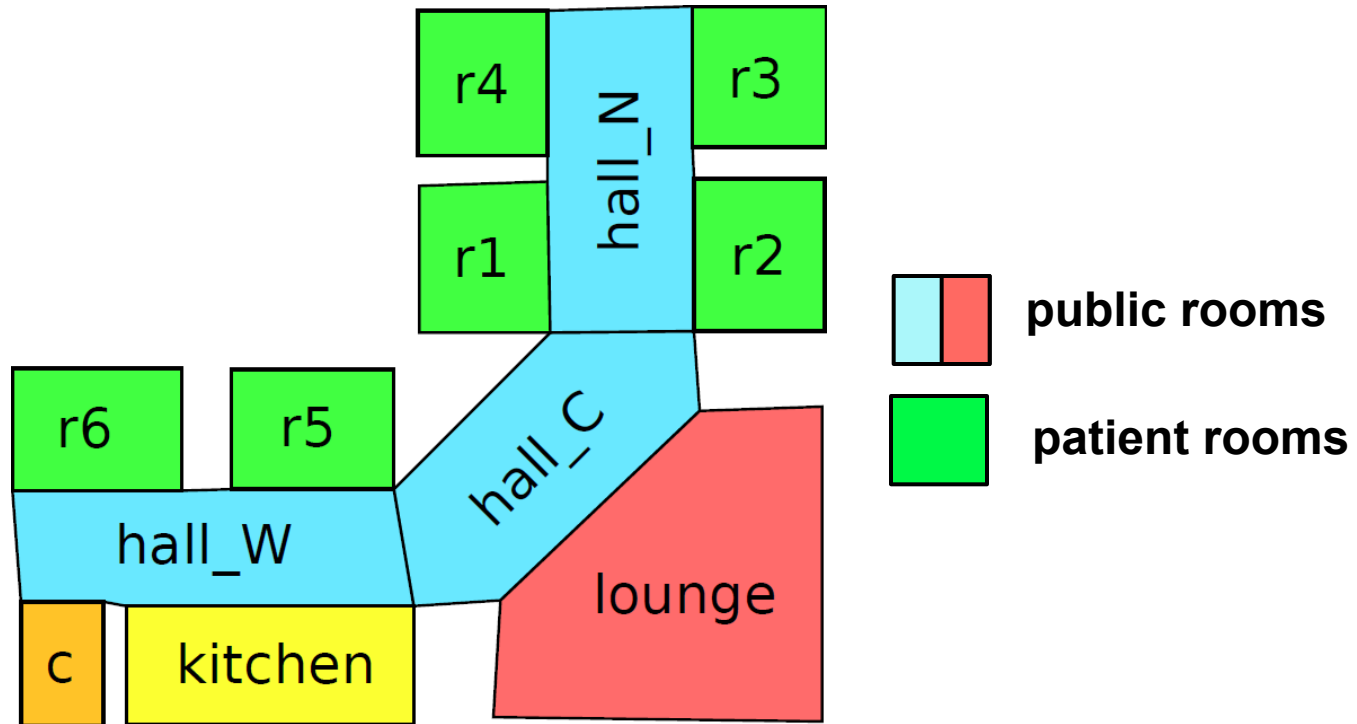
http://newsroom.ucla.edu/portal/ucla/artwork/4/5/7/9/4/245794/EVA_Robot_3-c.jpg

Example



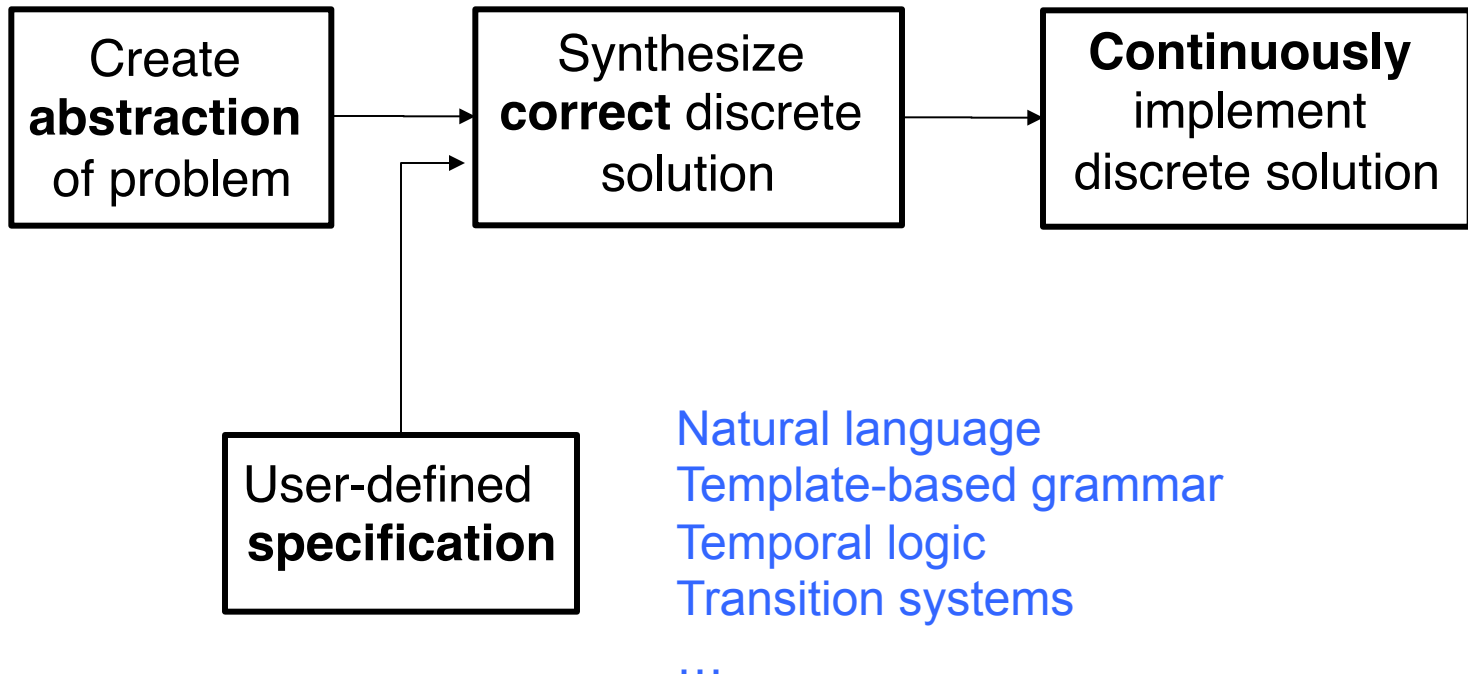
Carry meals from the kitchen to all patient rooms.

Example

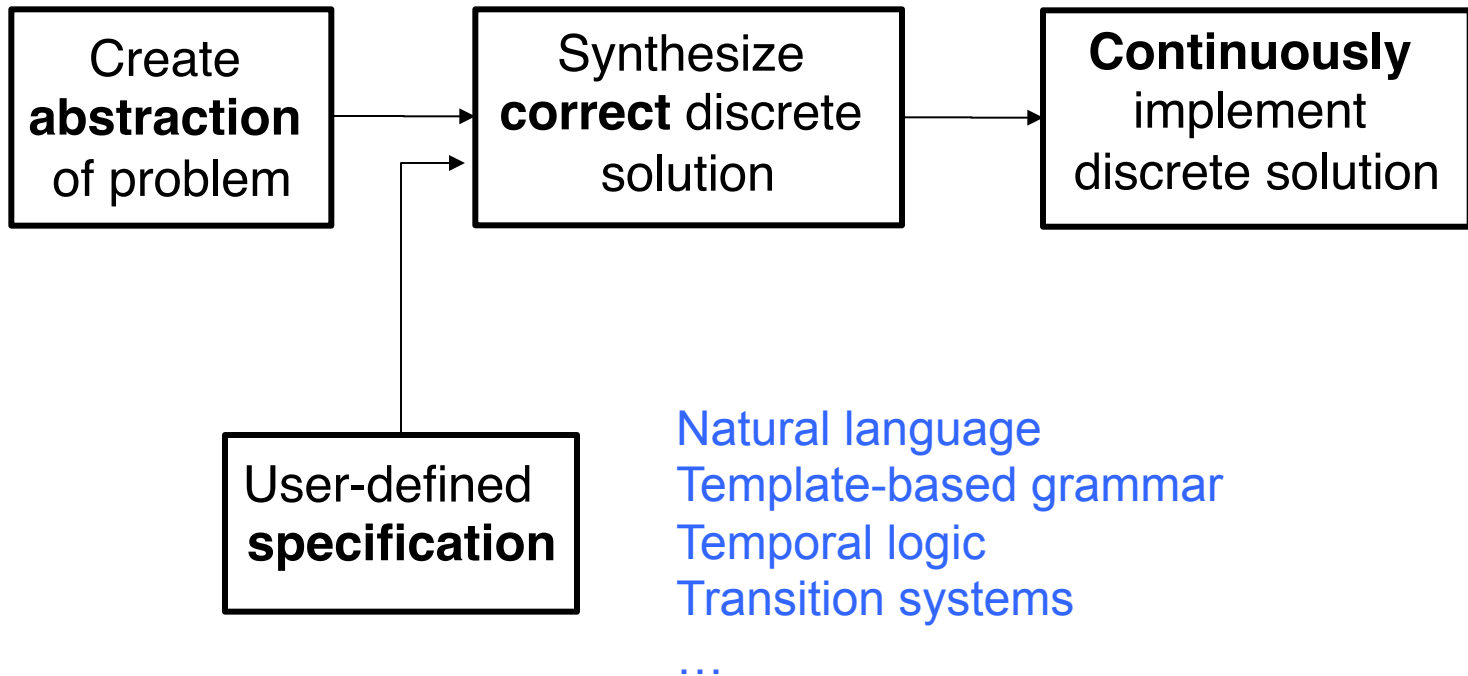


Start in the closet. Carry meals from the kitchen to all patient rooms. Don't go into any public rooms.

Formal Methods for High-Level Control

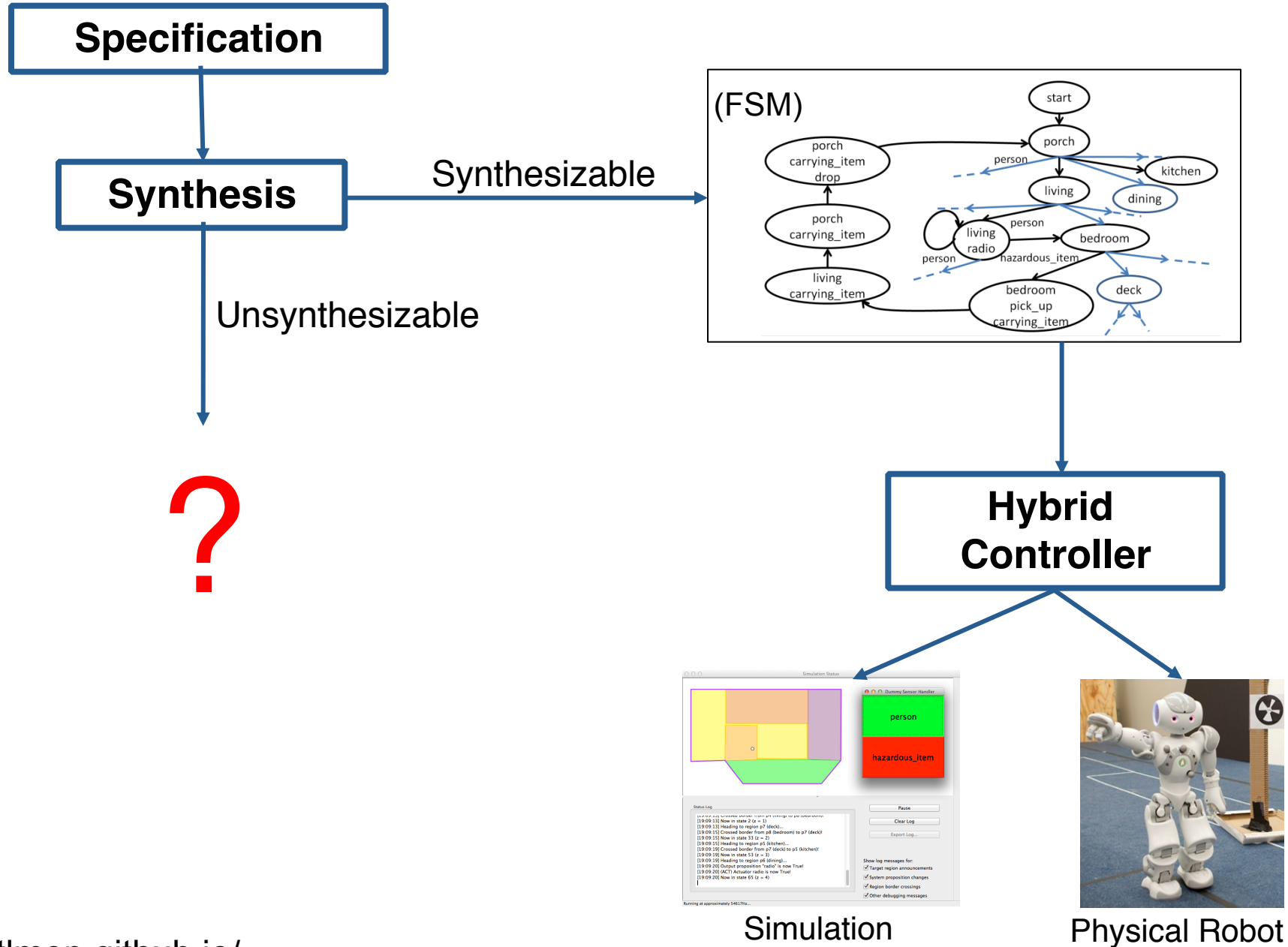


Formal Methods for High-Level Control

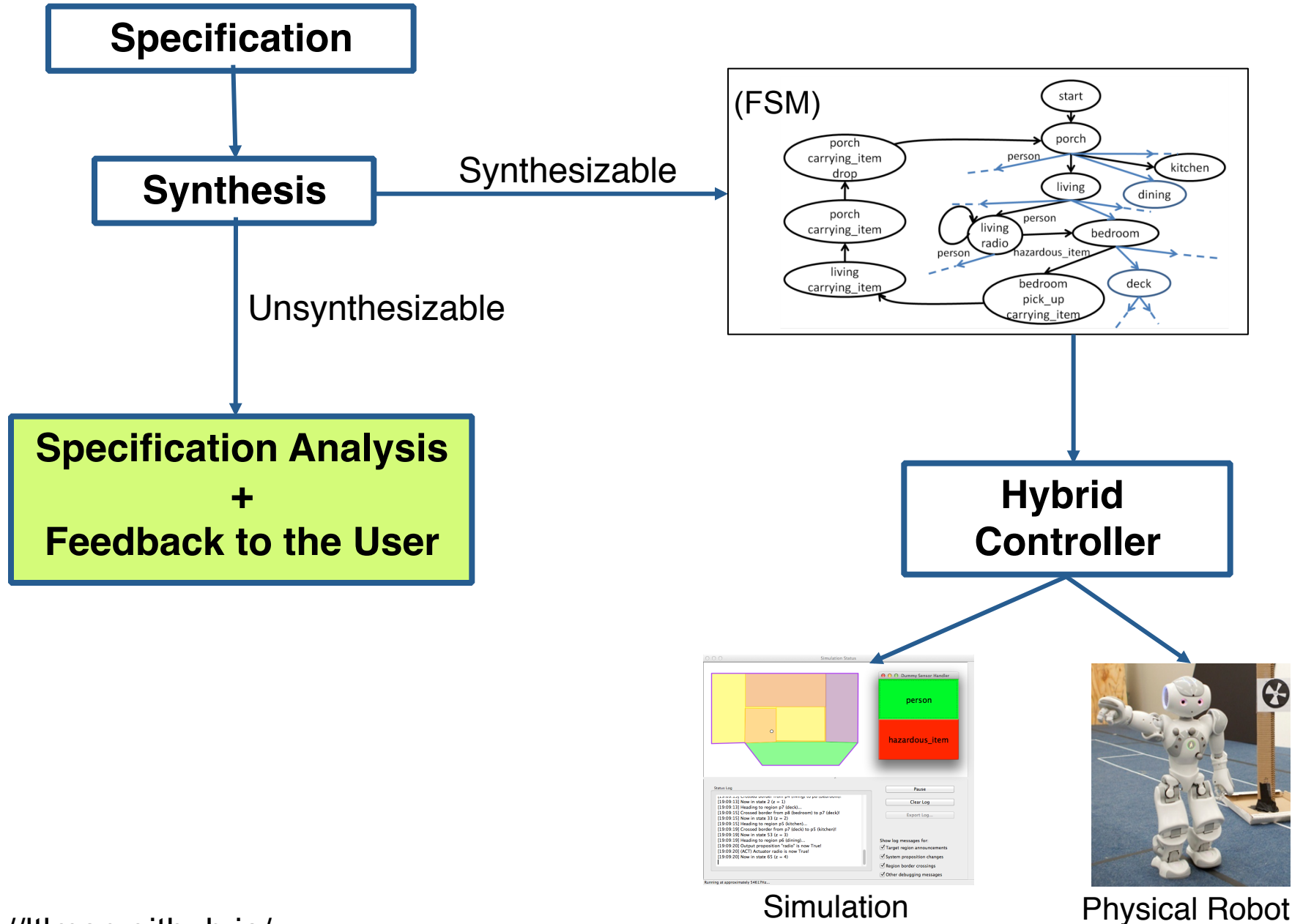


- Fainekos, Kress-Gazit and Pappas, ICRA 2005
- Kress-Gazit, Fainekos and Pappas, ICRA 2007
- Kloetzer and Belta, TAC 2008
- Karaman and Frazzoli, CDC 2009
- Bhatia, Kavraki and Vardi, ICRA 2010
- Wongpiromsarn, Topcu and Murray, HSCC 2010

Controller Synthesis Overview (LTLMoP* Toolkit)



Controller Synthesis Overview (LTLMoP* Toolkit)



Controller Synthesis Overview (LTLMoP* Toolkit)

Specification

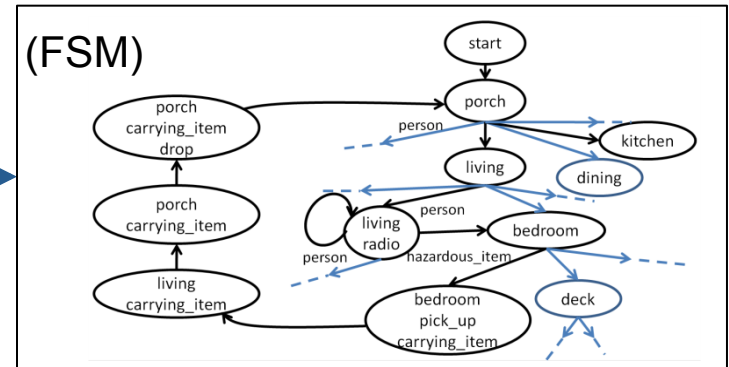
Synthesis

Synthesizable

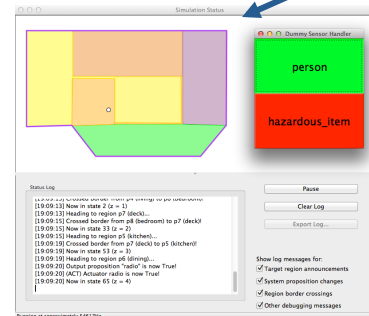
Unsynthesizable

Specification Analysis
+
Feedback to the User

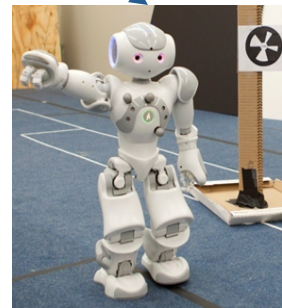
Change the
Specification



Hybrid
Controller



Simulation



Physical Robot

Controller Synthesis Overview (LTLMoP* Toolkit)

Specification

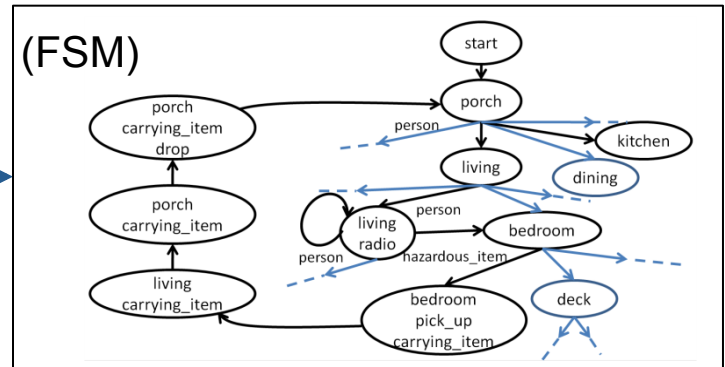
Synthesis

Synthesizable

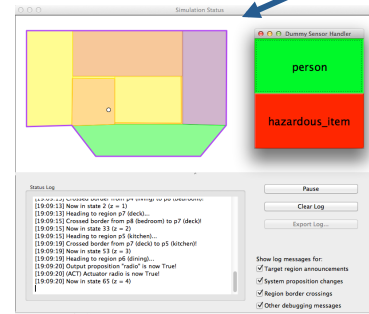
Unsynthesizable

Specification Analysis
+
Feedback to the User

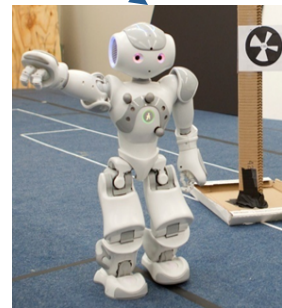
Change the
Specification



Hybrid
Controller



Simulation



Physical Robot

This talk

Form of Specification

$$\begin{aligned}\varphi &= (\varphi_e \Rightarrow \varphi_s) \\ &= \underbrace{(\varphi_e^i \wedge \varphi_e^t \wedge \varphi_e^g)}_{\text{Environment assumptions}} \Rightarrow \underbrace{(\varphi_s^i \wedge \varphi_s^t \wedge \varphi_s^g)}_{\text{Desired guarantees}}\end{aligned}$$

UNSYNTHESIZABLE

Unsatisfiable

Unrealizable

The diagram illustrates the form of a specification $\varphi = (\varphi_e \Rightarrow \varphi_s)$. It is expanded into $(\varphi_e^i \wedge \varphi_e^t \wedge \varphi_e^g \Rightarrow \varphi_s^i \wedge \varphi_s^t \wedge \varphi_s^g)$. The left side, $\varphi_e^i \wedge \varphi_e^t \wedge \varphi_e^g$, is labeled 'Environment assumptions' and 'UNSYNTHESIZABLE'. The right side, $\varphi_s^i \wedge \varphi_s^t \wedge \varphi_s^g$, is labeled 'Desired guarantees' and 'Unsatisfiable'. A bracket spanning both sides is labeled 'Unrealizable'.

Two levels of analysis:

- identify subformulas that contribute
- compute minimal subformula causing failure

Form of Specification

$$\begin{aligned}\varphi &= (\varphi_e \Rightarrow \varphi_s) \\ &= \underbrace{(\varphi_e^i \wedge \varphi_e^t \wedge \varphi_e^g)}_{\text{Environment assumptions}} \Rightarrow \underbrace{(\varphi_s^i \wedge \varphi_s^t \wedge \varphi_s^g)}_{\text{Desired guarantees}}\end{aligned}$$

UNSYNTHESIZABLE

Unsatisfiable

Unrealizable

The diagram illustrates the form of a specification $\varphi = (\varphi_e \Rightarrow \varphi_s)$. It is expanded into $(\varphi_e^i \wedge \varphi_e^t \wedge \varphi_e^g \Rightarrow \varphi_s^i \wedge \varphi_s^t \wedge \varphi_s^g)$. The left side, $\varphi_e^i \wedge \varphi_e^t \wedge \varphi_e^g$, is labeled 'Environment assumptions' in red. The right side, $\varphi_s^i \wedge \varphi_s^t \wedge \varphi_s^g$, is labeled 'Desired guarantees' in red. A blue bracket under 'UNSYNTHESIZABLE' spans the entire formula. Another blue bracket under 'Unsatisfiable' spans the 'Desired guarantees' part. A large blue bracket at the bottom, labeled 'Unrealizable', spans the entire formula.

Two levels of analysis:

- identify subformulas that contribute
- **compute minimal subformula causing failure**

Problem Statement

Highlight a MINIMAL cause of unsynthesizability

- Find a small subformula φ' of φ such that:
 - φ' is by itself unsynthesizable
(an unsynthesizable “core”)
 - every proper subformula of φ' is synthesizable

Linear Temporal Logic (LTL)

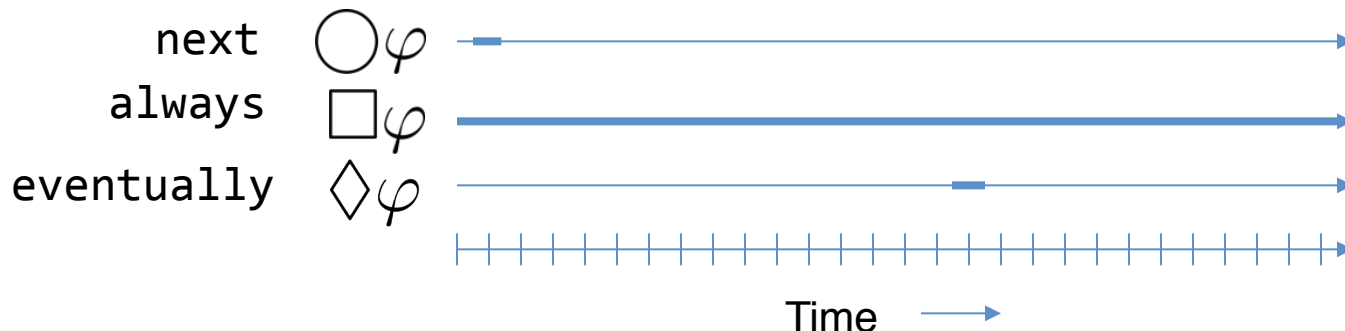
Syntax

AP is a set of atomic propositions, $\pi \in AP$

$$\varphi ::= \pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi\mathcal{U}\varphi$$

Semantics

$\sigma \models \varphi$: infinite sequence of truth assignments σ satisfies φ



Unsatisfiable Cores

via propositional satisfiability (SAT)

General Idea:

1. “Unroll” LTL specification to **some depth**
(encode as a propositional SAT problem)
2. Use off-the-shelf SAT solver to find **MUS**
3. Given a MUS, **map it back** to the LTL

Unsatisfiable Cores

via propositional satisfiability (SAT)

Unrolling the LTL specification:

- Fix unroll depth d
- Construct SAT instance
 - instantiate each atomic proposition for each time step from 0 to d
 - restrict the propositions at each time step

e.g. $\Box\varphi \rightarrow \varphi_0 \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \dots \wedge \varphi_d$

always wave $\rightarrow \text{wave}_1 \wedge \text{wave}_2 \wedge \dots \wedge \text{wave}_d$

Unsatisfiable Cores

via propositional satisfiability (SAT)

Unrolling the LTL specification:

- Fix unroll depth d
- Construct SAT instance

$$\Box\varphi \rightarrow \varphi_0 \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \dots \wedge \varphi_d$$

e.g. `always wave` $\rightarrow$ `wave1` $\wedge$ `wave2` $\wedge$... $\wedge$ `waved`

Unsatisfiable Cores

via propositional satisfiability (SAT)

Use off-the-shelf SAT solver to find MUS:

- Input – unrolled specification in CNF form
- Output – Subset of CNF clauses

e.g. PicoSAT*

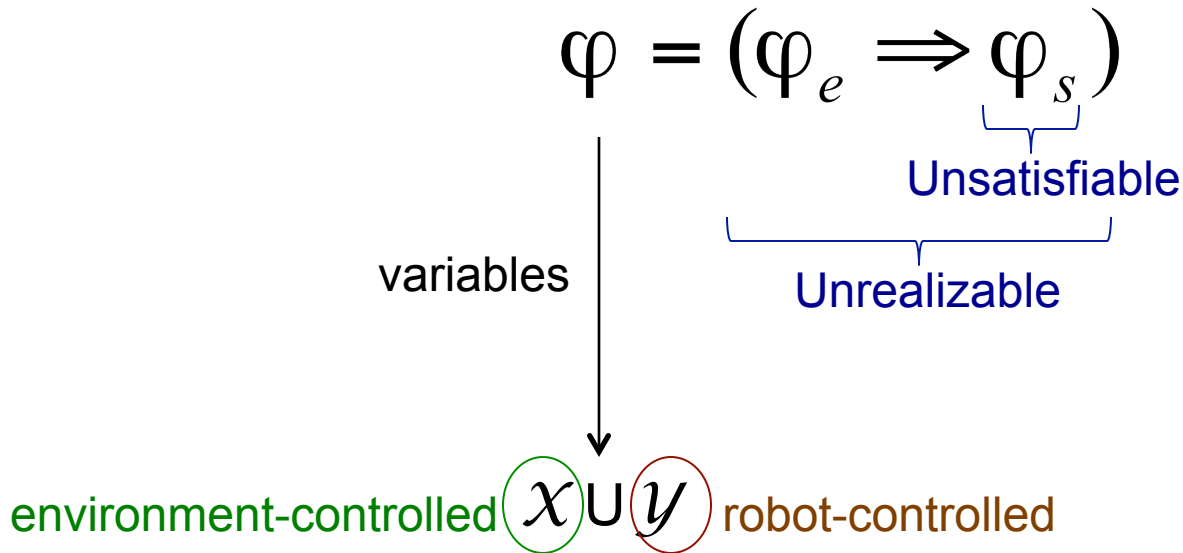
Unsatisfiable Cores

via propositional satisfiability (SAT)

Given an MUS, **map it back** to the LTL:

1. Track origin of each CNF clause
2. Depth of unrolling determines “core” found

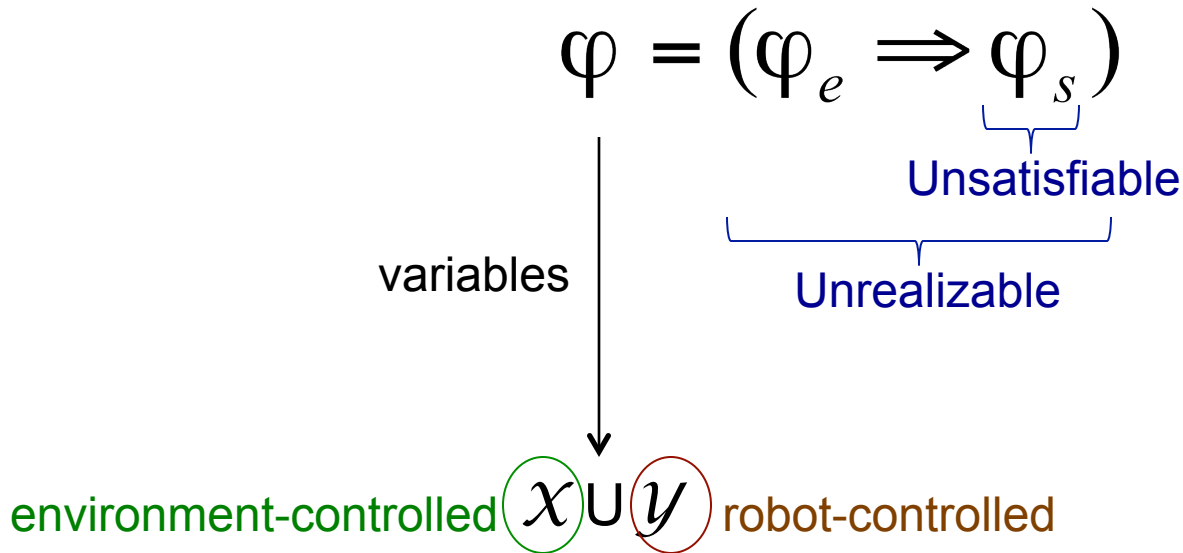
Unrealizable “Cores”



Unsatisfiability: no assignment to $\mathcal{X} \cup \mathcal{Y}$ satisfies φ

Unrealizability: exists assignment to $\mathcal{X}$ such that no assignment to $\mathcal{Y}$ satisfies φ

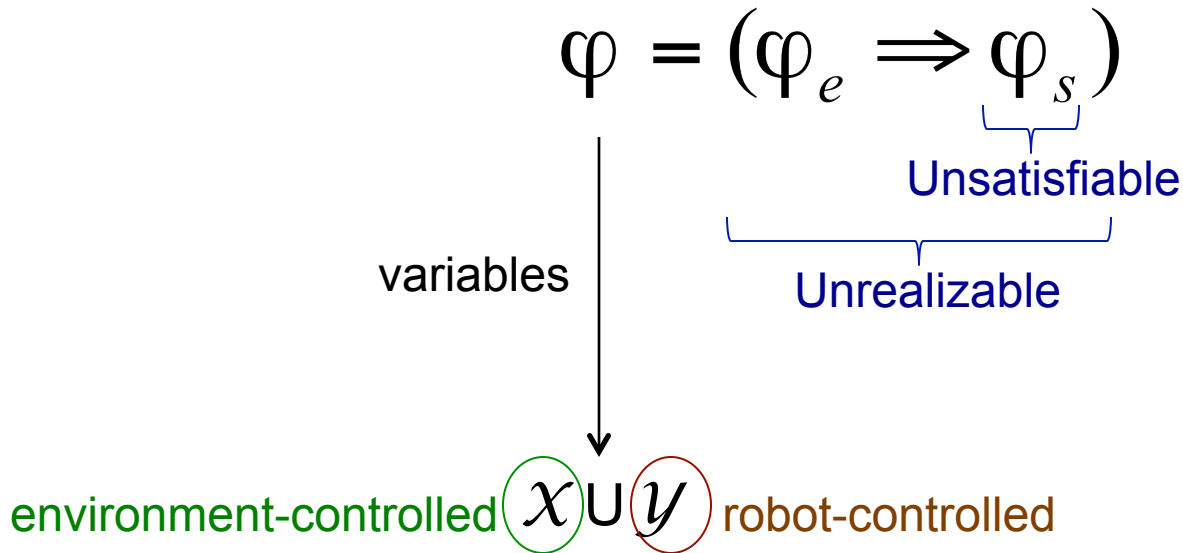
Unrealizable “Cores”



Can we still use SAT-based techniques?

Yes, but we need to restrict the environment variables $\mathcal{X}$ in the “right” way

Unrealizable “Cores”



Counterstrategy + SAT-based techniques:

1. Unroll the specification as before
2. Restrict inputs according to **environment counterstrategy**
3. Compute MUS of resulting SAT formula

Unrealizable “Cores”

Two types of unrealizability

- *Deadlock*: a state with no next robot move
Blocking states in the counterstrategy graph
(states with no out-transitions)
- *Livelock*: cycle of states that do not satisfy the goals
Cycles in the counterstrategy graph
Unroll to a certain depth to restrict environment

Unrealizable “Cores”

Iterated realizability tests:

1. Remove conjunct i and test realizability
2. If the specification is still unrealizable, leave it out, otherwise add it back in
3. Repeat for $i++$

Remaining conjuncts form an unrealizable core

Unrealizable “Cores”

```
Specification Editor - firefighting.spec
File Edit Run Debug Help
1 # Initial conditions
2 Env starts with false
3 ▶ Robot starts in porch with false
4
5 # Assumptions about the environment
6 If you were in porch then do not hazardous
7
8 # Define robot safety including how to pick up
9 ▶ Do pick_up if and only if you are sensing
10 you are not activating carrying_item
11 ▶ carrying_item is set on pick_up and reset
12 ▶ Do drop if and only if you are in porch and
13 activating carrying_item
14
15 ▶ If you did not activate carrying_item then
16
17 # Define when and how to radio
18 ▶ Do radio if and only if you are sensing person
19 ▶ If you are activating radio or you were activating
20 stay there
21 Always extinguish
22
23 # Patrol goals
24 Group rooms is living, bedroom, deck, kitchen, dining
25 If you are not activating carrying_item and you are not
26 activating radio then visit all rooms
27 if you are activating carrying_item and you are not
28 activating radio then visit porch
```

Compiler Log LTL Output Workspace Decomposition

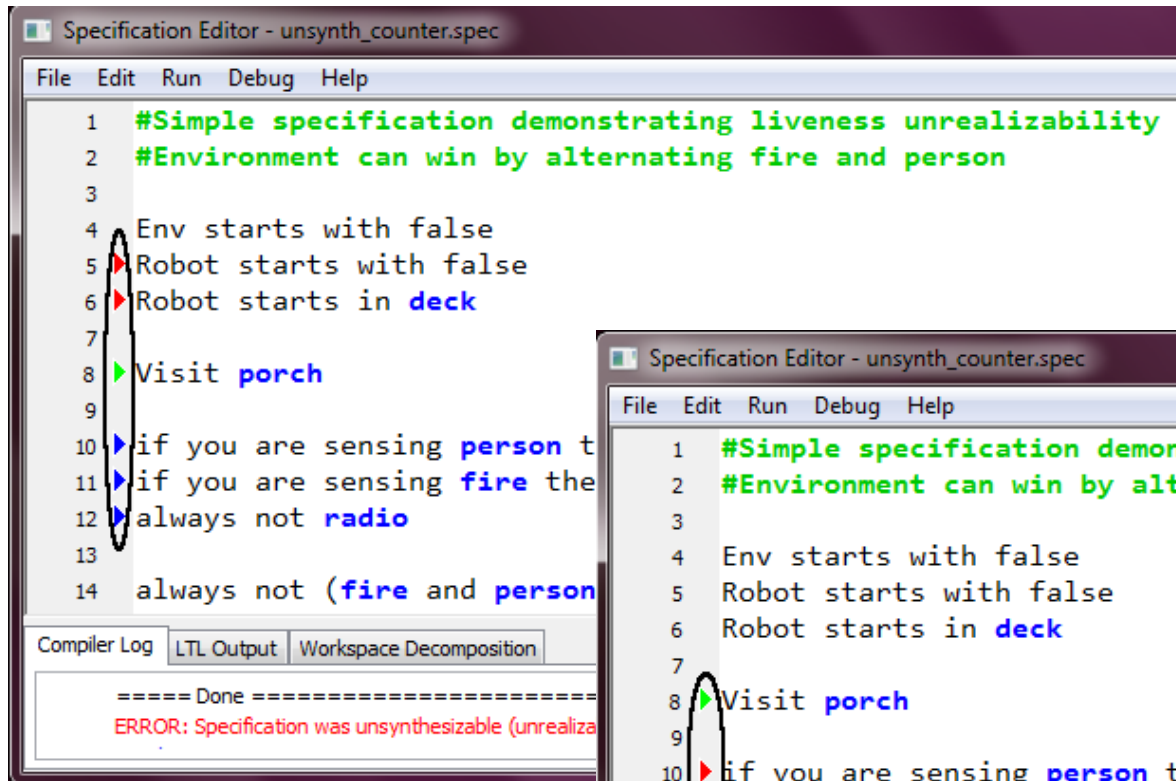
===== Done =====
ERROR: Specification was unsynthesizable (unrealizable/unsatisfiable) for instantaneous actions.

```
Specification Editor - firefighting.spec
File Edit Run Debug Help
1 # Initial conditions
2 Env starts with false
3 Robot starts in porch with false
4
5 # Assumptions about the environment
6 If you were in porch then do not hazardous_item
7
8 # Define robot safety including how to pick up
9 Do pick_up if and only if you are sensing hazardous_item and
10 you are not activating carrying_item
11 carrying_item is set on pick_up and reset on drop
12 Do drop if and only if you are in porch and you are
13 activating carrying_item
14
15 ▶ If you did not activate carrying_item then always not porch
16
17 # Define when and how to radio
18 ▶ Do radio if and only if you are sensing person
19 ▶ If you are activating radio or you were activating radio then
20 stay there
21 Always extinguish
22
23 # Patrol goals
24 Group rooms is living, bedroom, deck, kitchen, dining
25 If you are not activating carrying_item and you are not
26 activating radio then visit all rooms
27 if you are activating carrying_item and you are not
28 activating radio then visit porch
```

Compiler Log LTL Output Workspace Decomposition

===== Done =====
ERROR: Specification was unsynthesizable (unrealizable/unsatisfiable) for instantaneous actions.

Unrealizable “Cores”

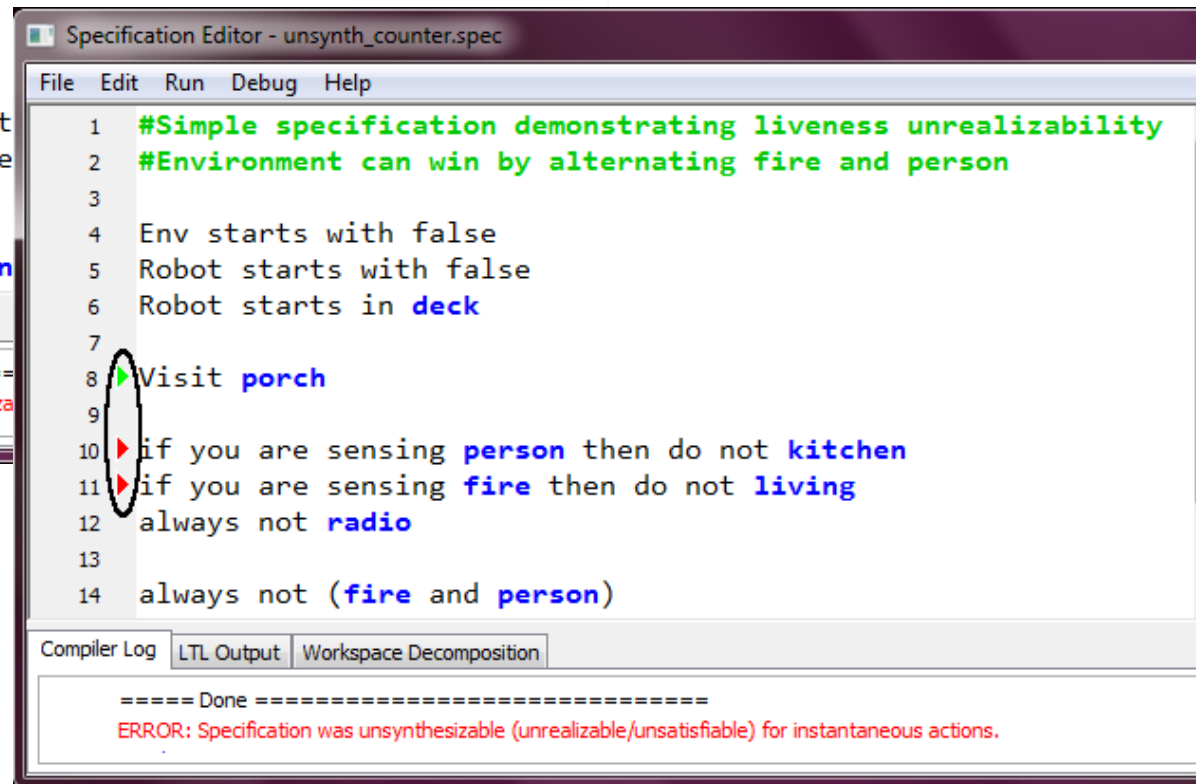


Specification Editor - unsynth_counter.spec

```
File Edit Run Debug Help
1 #Simple specification demonstrating liveness unrealizability
2 #Environment can win by alternating fire and person
3
4 Env starts with false
5 ▶ Robot starts with false
6 ▶ Robot starts in deck
7
8 ▶ Visit porch
9
10 ▶ if you are sensing person then do not kitchen
11 ▶ if you are sensing fire then do not living
12 ▶ always not radio
13
14 always not (fire and person)
```

Compiler Log LTL Output Workspace Decomposition

===== Done =====
ERROR: Specification was unsynthesizable (unrealizable/ununsatisfiable) for instantaneous actions.



Specification Editor - unsynth_counter.spec

```
File Edit Run Debug Help
1 #Simple specification demonstrating liveness unrealizability
2 #Environment can win by alternating fire and person
3
4 Env starts with false
5 Robot starts with false
6 Robot starts in deck
7
8 ▶ Visit porch
9
10 ▶ if you are sensing person then do not kitchen
11 ▶ if you are sensing fire then do not living
12 always not radio
13
14 always not (fire and person)
```

Compiler Log LTL Output Workspace Decomposition

===== Done =====
ERROR: Specification was unsynthesizable (unrealizable/ununsatisfiable) for instantaneous actions.

Analysis Output:

The problematic goal is 'Carry meals from the kitchen to all patient rooms.'. The system cannot achieve the sub-goal "Deliver 'meal' to 'r1'.".

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'.".

No further analysis available.

SLURP Traceback:

- ▶ Start in the closet.
- ▼ Carry meals from the kitchen to all patient rooms.
 - ▼ Action: 'carry', Argument: 'meal', Source: 'kitchen', Destination: 'rooms'
 - ▶ Nothing is carried or delivered at the start.
 - ▶ Only pick up if you can carry more.
 - ▶ Only drop if you are carrying something.
 - ▶ Stay where you are when picking up and dropping.
 - ▶ Pick up 'meal' in 'kitchen'.
 - ▼ Deliver 'meal' to 'r1'.

```
(([]((next(s.mem_deliver_r1) <-> (s.mem_deliver_r1 | (next(s.r1) & next(s.drop))))))
```

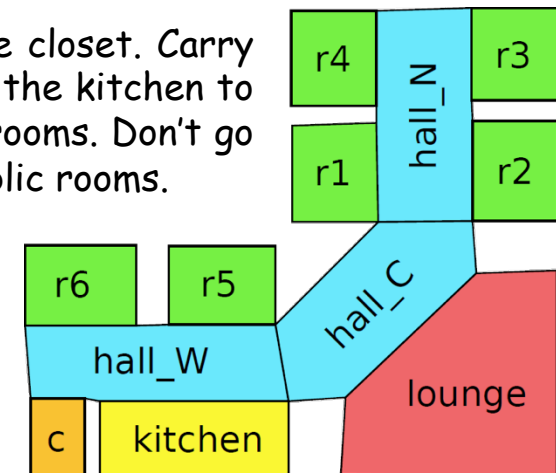
```
(([]<>(s.mem_deliver_r1))
```

- ▶ Deliver 'meal' to 'r2'.
- ▶ Deliver 'meal' to 'r3'.
- ▶ Deliver 'meal' to 'r4'.
- ▶ Deliver 'meal' to 'r5'.
- ▶ Deliver 'meal' to 'r6'.
- ▼ Don't go into any public rooms.
 - ▼ Action: do not 'go', Location: 'rooms'
 - ▼ Do not go to 'hall_c'.

```
(([](!s.hall_c))
```

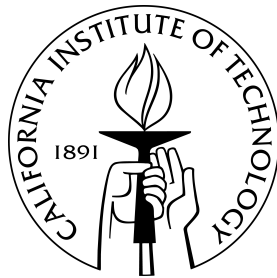
- ▶ The robot does not begin in 'hall_c'.
- ▶ Do not go to 'lounge'.
- ▶ The robot does not begin in 'lounge'.

Start in the closet. Carry meals from the kitchen to all patient rooms. Don't go into any public rooms.



Towards Minimal Explanations of Unsynthesizability for High-Level Robot Behaviors

Vasu Raman
vasu@caltech.edu



Hadas Kress-Gazit
hadaskg@cornell.edu

