Avoiding Forgetfulness: Structured English Specifications for High-Level Robot Control with Implicit Memory

Vasu Raman
Bingxin Xu
Hadas Kress-Gazit
(presented by Cameron Finucane)

Cornell University
Motivation

- High-level robot tasks

Patrol the aisles!
Motivation

• High-level robot tasks → No guarantees!

Patrol the aisles!
Motivation

- High-level robot tasks → No guarantees!
- Synthesis from formal specifications

\[
\begin{align*}
&[]<>r1 \\
&[]<>r2 \\
&[]<>r3 \\
&[]<>r4!
\end{align*}
\]
Motivation

- High-level robot tasks → No guarantees!
- Synthesis from formal specifications → Unintuitive!

\[
\begin{align*}
&[]<>r1 \\
&[]<>r2 \\
&[]<>r3 \\
&[]<>r4
\end{align*}
\]
Motivation

- High-level robot tasks → No guarantees!
- Synthesis from formal specifications → Unintuitive!
- Intuitive human interface

Visit all corners
Linear Temporal Logic Mission Planning Toolkit (LTLMoP)

- Structured English input → Correct robot control
- Grammar allows:
  - Conditionals (“if”, “if and only if”)
  - Locative prepositions (“between”, “near”)
  - Region quantifiers (“any”, “all”)
  - Goals (“visit all checkpoints”)
  - Safety requirements (“avoid the kitchen”)

LTLMoP – behind the scenes

• Structured English input
  → Linear Temporal Logic formulas

• Tied tightly to underlying formalism (LTL fragment)
Linear Temporal Logic

• Syntax:

\[ \varphi ::= \pi \mid \neg \varphi \mid \varphi \lor \varphi \mid \Diamond \varphi \mid \Box \varphi \mid \vartriangle \varphi \]

• Temporal operators:

\[ \Diamond \varphi \quad \text{next step} \]
\[ \Box \varphi \quad \text{always} \]
\[ \vartriangle \varphi \quad \text{eventually} \]
Example

“If you are given an order then go to the kitchen”
Example

“If you are given an order then go to the kitchen”
What is the right LTL formula to capture this?
Example

“If you are given an order then go to the kitchen”
What is the right LTL formula to capture this?

• Initial guess (direct translation):

  If you are sensing order then visit kitchen
  \[ \Box \Diamond (\pi_{order} \implies \pi_{kitchen}) \]
Example

“If you are given an order then go to the kitchen”

Implicit memory:

• Need to remember an order was received
Example

“If you are given an order then go to the kitchen”
What is the right LTL formula to capture this?

• Specification for desired behavior

  $m_{\text{order}}$ is set on order and never reset
  
  $\square(\Diamond m_{\text{order}} \iff (\Diamond \pi_{\text{order}} \lor m_{\text{order}}))$

  If you are activating $m_{\text{order}}$ then visit kitchen

  $\square \Diamond (m_{\text{order}} \Rightarrow \pi_{\text{kitchen}})$
This paper

- Allow users to specify tasks that include event memory
- **Automatically** define “memory propositions”
Memory Propositions

- Implicit
  - Not defined by user
  - Only appear in LTL, not structured English
- Respond to the explicitly specified event
  - $m_\phi$ responds to event $\phi$
- Example:

  \[
  \Box(\Box m_\phi \iff (\Box \phi \lor m_\phi))
  \]
## Grammar for implicit memory

<table>
<thead>
<tr>
<th>Type</th>
<th>What to remember?</th>
<th>Structured English $(S)$</th>
<th>LTL $(\mathcal{M}, \Phi)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Condition has happened</td>
<td>Once $\Theta_{\text{cond}}$ then $\Theta_{\text{req-safe}}$ from now on</td>
<td>$\square(m_{\text{cond}} \Rightarrow \phi_{\text{req-safe}}) \land$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>After $\Theta_{\text{cond}}$ then $\Theta_{\text{req-live}}$ repeatedly</td>
<td>$\square((\square m_{\text{cond}} \Leftrightarrow (\square \phi_{\text{cond}} \lor m_{\text{cond}}))$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\square(m_{\text{cond}} \Rightarrow \phi_{\text{req-live}}) \land$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\square((\square m_{\text{cond}} \Leftrightarrow (\square \phi_{\text{cond}} \lor m_{\text{cond}}))$</td>
</tr>
<tr>
<td>2</td>
<td>Requirement has happened</td>
<td>$\Theta_{\text{req}}$ (at least once)</td>
<td>$\square \diamond (m_{\text{req}})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\square((\square \phi_{\text{req}} \lor m_{\text{cond}}))$</td>
</tr>
<tr>
<td>3</td>
<td>Requirement has happened under certain condition</td>
<td>While $\Theta_{\text{cond}}$ then $\Theta_{\text{req}}$ (at least once)</td>
<td>$\Delta(\phi_{\text{cond}} \Rightarrow \phi_{\text{cond}} \land \phi_{\text{req}})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\square((\square \phi_{\text{cond}} \lor \phi_{\text{cond}}))$</td>
</tr>
<tr>
<td>4</td>
<td>Memo is set on $\Theta_1$ and reset on $\Theta_2$</td>
<td>After/once $\Theta_1$ then $\Theta_{\text{req}}$ until $\Theta_2$</td>
<td>$\Delta(m_{\phi_1} \phi_2 \Rightarrow \phi_{\text{req}})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\square((\square \phi_{\text{req}} \lor \phi_{\text{cond}}))$</td>
</tr>
<tr>
<td>*1</td>
<td>'Only'+cond</td>
<td>Only once $\Theta_{\text{cond}}$ then $\Theta_{\text{req-safe}}$ from now on</td>
<td>LTL in Type 1 + $\square((\square \phi_{\text{cond}} \lor \phi_{\text{cond}})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Only after $\Theta_{\text{cond}}$ then $\Theta_{\text{req-live}}$ repeatedly</td>
<td>$\square((\square \phi_{\text{cond}} \lor \phi_{\text{cond}}))$</td>
</tr>
<tr>
<td>*2</td>
<td>requirement + 'only once'</td>
<td>Eventually $\Theta_{\text{req-live}}$ only once</td>
<td>LTL in Type 2 + $\square((\square \phi_{\text{cond}} \lor \phi_{\text{cond}})$</td>
</tr>
<tr>
<td>*3</td>
<td>requirement under condition + 'only once'</td>
<td>If $\Theta_{\text{cond}}$ then eventually $\Theta_{\text{req-live}}$ only once</td>
<td>LTL in Type 3 + $\square((\square \phi_{\text{cond}} \lor \phi_{\text{cond}})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If $\Theta_{\text{cond}}$ then $\Theta_{\text{req-safe}}$ only once</td>
<td>$\square((\square \phi_{\text{cond}} \lor \phi_{\text{cond}})$</td>
</tr>
<tr>
<td>*4</td>
<td>Memo is self-reset when the requirement is met</td>
<td>After each time $\Theta_{\text{cond}}$, $\Theta_{\text{req}}$ (at least once)</td>
<td>$\Delta(m_{\phi_{\text{cond}}} \phi_{\text{req}} \Rightarrow \phi_{\text{req}})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\square((\square \phi_{\text{cond}} \lor m_{\phi_{\text{cond}}})$</td>
<td></td>
</tr>
<tr>
<td>*5</td>
<td>Condition-Requirement memos on both sides</td>
<td>After the first time $\Theta_{\text{cond}}$, $\Theta_{\text{req}}$ (at least once)</td>
<td>$\Delta(m_{\phi_{\text{cond}}} \Rightarrow m_{\phi_{\text{cond}}})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\square((\square \phi_{\text{req}} \land m_{\phi_{\text{cond}}})$</td>
<td>$\lor \phi_{\text{req}})$</td>
</tr>
</tbody>
</table>
After the first time $\Theta_{\text{cond}},$ $\Theta_{\text{req}}$ (at least once)

\[
\Delta(m_{\Phi_{\text{cond}}} \Rightarrow m_{\Phi_{\text{cond}} - \Phi_{\text{req}}})
\]

<table>
<thead>
<tr>
<th>Type</th>
<th>What to remember?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Condition</td>
</tr>
<tr>
<td>2</td>
<td>Requirement</td>
</tr>
<tr>
<td>3</td>
<td>Requirement under condition</td>
</tr>
<tr>
<td>4</td>
<td>Memo is self-set on both sides</td>
</tr>
<tr>
<td>5</td>
<td>Condition-Requirement</td>
</tr>
</tbody>
</table>

*1 'Only' + cond Only once
*2 requirement + 'only once'
*3 requirement under condition
*4 Memo is self-set when the requirement is met
*5 Condition-Requirement
Example: Robot Waiter

- **First** go to the check-in desk
- Meet the **first** truck at the loading dock, but **ignore all following** trucks
- **When** customers arrive, move between the three dining rooms **until** accepting an order
- **Each time** an order is made, go to the kitchen
Task 2

Meet the **first** truck at the loading dock, but **ignore all following** trucks

- **Structured English:**
  After the first time you have sensed truck, go to loading_dock

- **Grammar:**
  After the first time $\Theta_{cond}, \Theta_{req}$ (at least once)

- **LTL:**
  $\Box(\Box m_{truck} \iff (\Box \pi_{truck} \lor m_{truck}))$
  $\land \Box(\Box m_{truck\_dock} \iff ((\Box \pi_{dock} \land \Box m_{truck}) \lor m_{truck\_dock}))$
  $\land \Box \Diamond (m_{truck} \Rightarrow m_{truck\_dock})$
Task 3

When customers arrive, move between the three dining rooms until accepting an order.

• Structured English:

   After you have sensed customer then visit all dining_rooms until you are sensing order.

• Grammar: After/once $\Theta_1$ then $\Theta_{req}$ until $\Theta_2$

• LTL: $\Box(m_{cust\_no\_order} \Leftrightarrow ((\Box \pi_{cust} \lor m_{cust\_no\_order}) \land \neg \Box \pi_{order}))$

   $\land \land_{i=1,2,3} \Box \Diamond (m_{cust\_no\_order} \Rightarrow \pi_{r_i})$
Complete Specification

Old Grammar

- Do memo_check_in if and only if you are in check_in_desk or you were activating memo_check_in
- Repeatedly visit memo_check_in
- Do memo_truck if and only if you are sensing truck or you were activating memo_truck
- Do memo_dock if and only if you are in loading_dock or you were activating memo_dock
- If you are activating memo_truck then visit memo_dock
- Do memo_customer if and only if (you are sensing customer or you were activating memo_customer) and you are not sensing order
- Group dining_rooms is room1, room2, room3
- If you are activating memo_dock then visit all dining_rooms
- Do memo_order if and only if (you are sensing order or you were activating memo_order) and you are not in kitchen
- If you are activating memo_order then visit kitchen

New Grammar

- Go to check_in_desk.
- After the first time you have sensed truck, go to loading_dock.
- Group dining_rooms is room1, room2, room3.
- After you have sensed customer then visit all dining_rooms until you are sensing order.
- After each time you have sensed order, go to kitchen.

Word Count: 121

Word Count: 44
Conclusions

• Enriched Structured English grammar for memory
  • Specifications translate to LTL
  • Automatic creation of memory propositions

• Accommodates several types of events

• More concise, intuitive specifications
Avoiding Forgetfulness: Structured English Specifications for High-Level Robot Control with Implicit Memory

Vasu Raman (vraman@cs.cornell.edu)
Bingxin Xu (bx38@cornell.edu)
Hadas Kress-Gazit (hadaskg@cornell.edu)
Cameron Finucane (cpf37@cornell.edu)

Cornell University

LTLMoP: http://ltlmop.github.com/ (GPL)