Natural Language is the Future!

Reasoning with user inputs

User → Constrained Natural Language → More precise parser for end users → “Easier to use & faster”

Natural Language → Formally Specified Language → Formal semantics → Inputs to reasoning engines

User → Re LS5 logicLangEngines

Linear-time Temporal Logic

A wonderful formal language to describe control rules.

- button → X(red U (elevator A doors!))
- G(weekdays A 7am) → (coffee)

Always make coffee at 7am on weekdays.

- G(red → X green)
- G(request → F acknowledgement)

Once the traffic light is red, the light cannot become green immediately after. Every request will eventually receive an acknowledgement.

- G(message → ¬sent U acknowledgement)

When a message is sent, an acknowledgement will eventually be returned, and the message will not be marked as sent before an acknowledgement is returned.

NL2LTL - API overview

<table>
<thead>
<tr>
<th>Translate</th>
<th>Template</th>
<th>NLU Engines</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLARE</td>
<td>+</td>
<td>RASA</td>
<td>WA +</td>
</tr>
<tr>
<td>NLU</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Future is the Past!

If you can, go with the Past

User → Natural Language → NL2LTL: an English to LTL converter → Linear-time Temporal Logic → Automated Planning

Linear-time Temporal Logic

Before achieving the task, the agent was not in the room anymore since the room was sanitized.

goal A Y(condition):

Enforce the agent to achieve the goal after some cooking has been made.

H(green → Y yellow):

Every time the traffic light is green, it has always been preceded by the yellow light.

H(ticket → Y(bus S ticket)):

The agent has always paid the ticket before getting the bus.

The Envisioned Product Impact

Alternative NLP processing pipeline: Stacked parsers