Linking Natural Language to Action

Hadas Kress-Gazit

Cornell University

Jan M. Allbeck

George Mason University

Advisors: George Pappas and Norm Badler

SUBTLE MURI

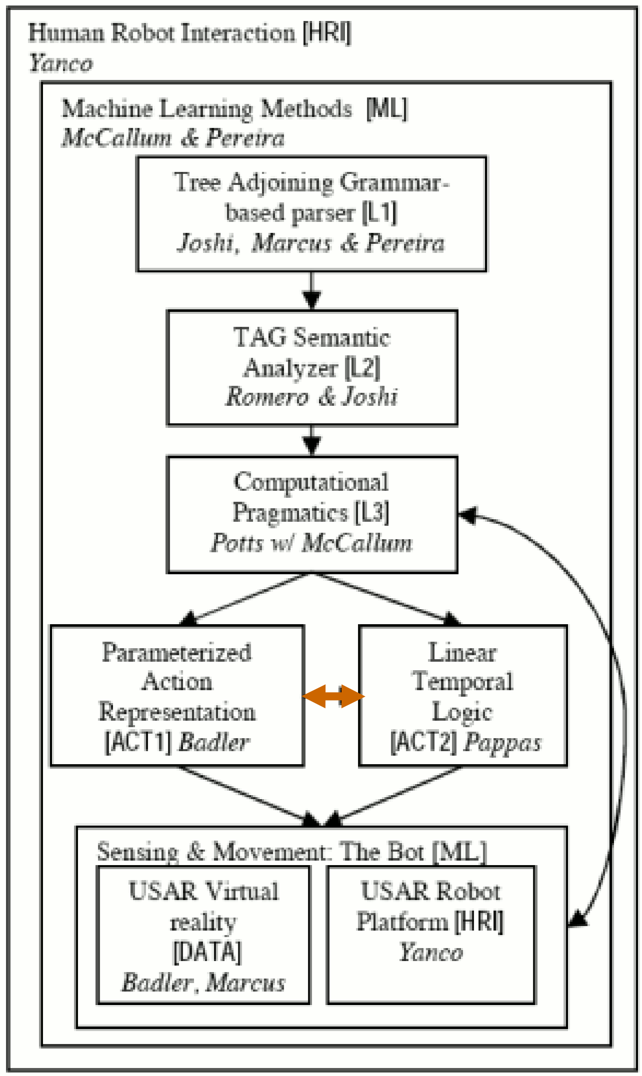


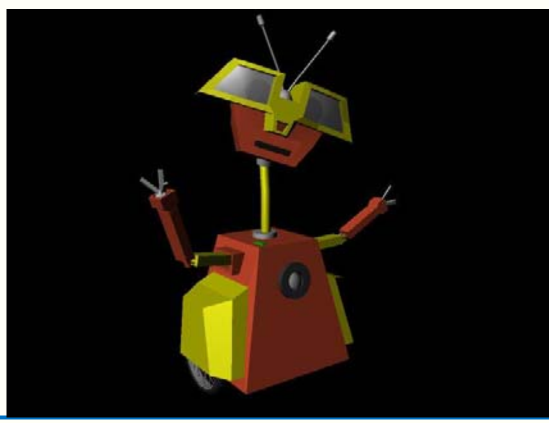
Figure 1: Team Structure

LTL and PAR Integration

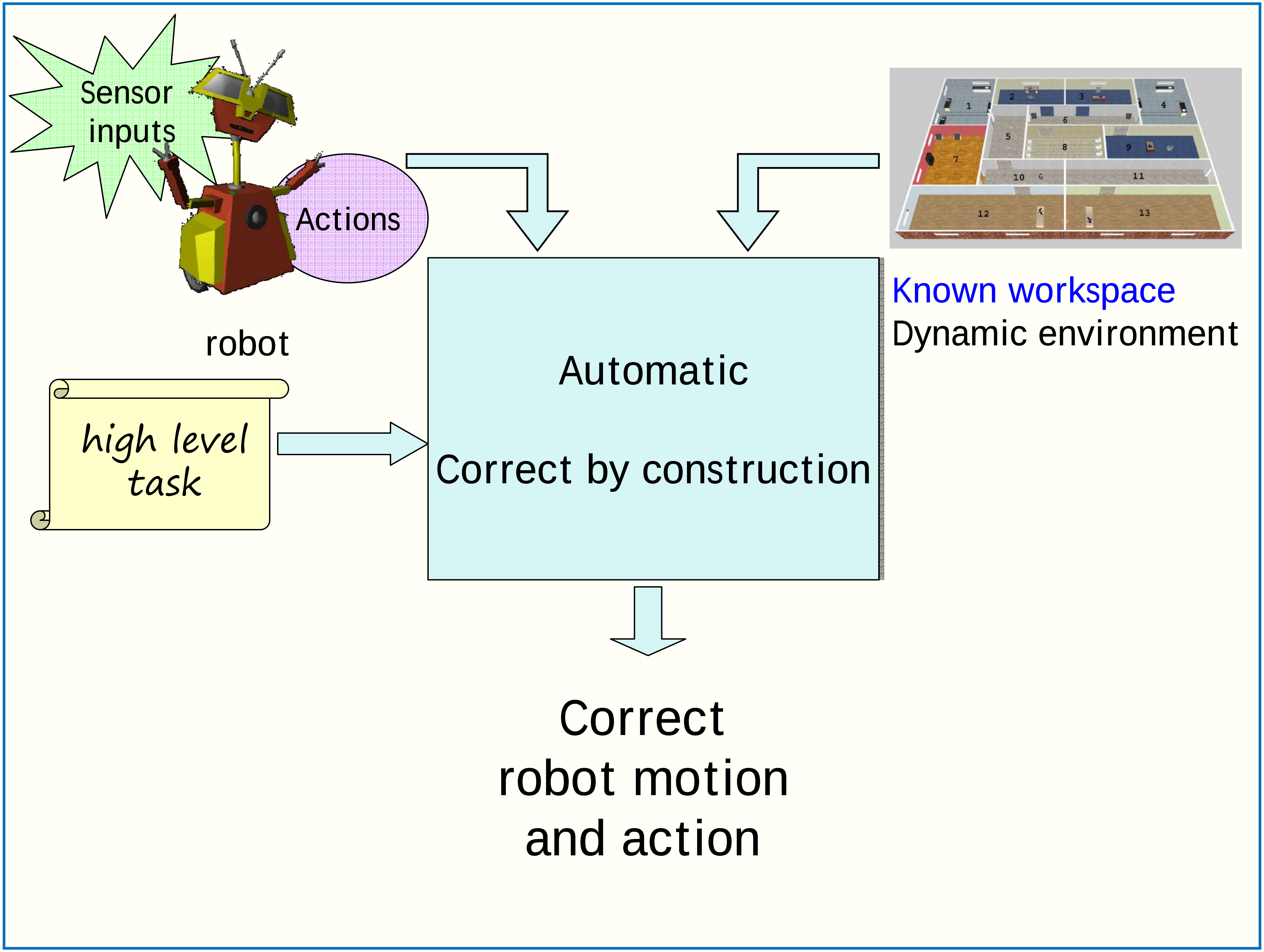
- Pragmatics->**PAR->LTL->PAR**
- Commands in the form of PARs will be instantiated from pragmatics.
- If the command requires planning (e.g. *search*), then LTL is called.
- LTL automatically and verifiably composes controllers that satisfy high level task specifications.
- PAR can then be used to fill in parameters of the actions and for simulation.
- Additionally, PAR provides LTL with precepts of the environment that produce state transitions in the LTL automaton.

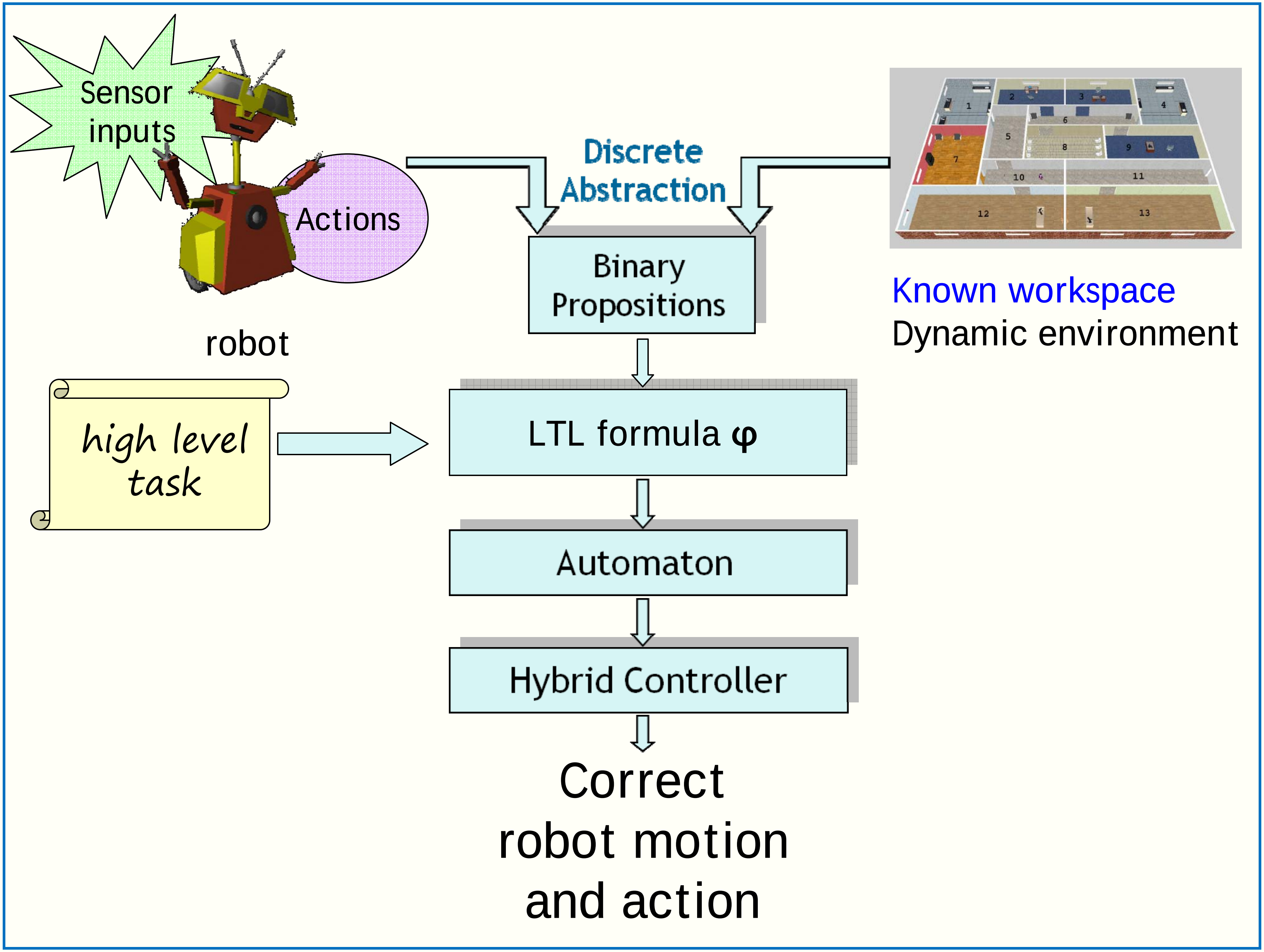
Example Mission

- Murray starts in room 11.
- "Search rooms 1,2,3 and 4. If you see a dead body, abandon the search and go to room 11. If you see a bomb, pick it up and take it to room 13 and then resume the search."



"Search rooms 1,2,3 and 4. If you see a dead body, abandon the search and go to room 11. If you see a bomb, pick it up and take it to room 13 and then resume the search."


Sensor inputs
Actions
robot
Known workspace
Dynamic environment
high level task
Automatic
Correct by construction
Correct robot motion and action



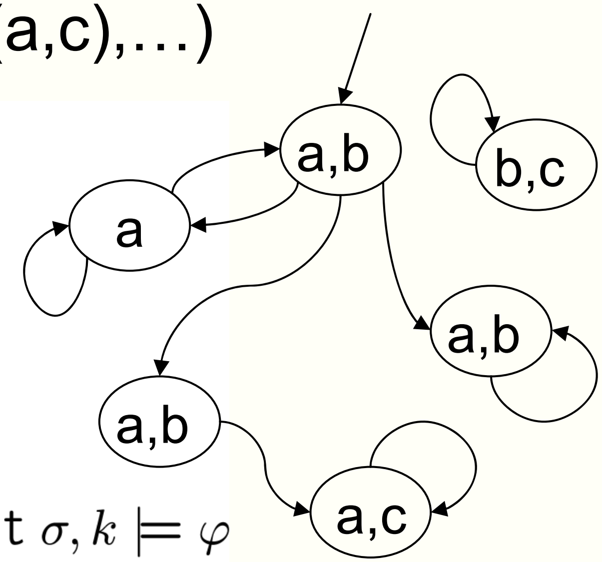
Sensor inputs
Actions
robot
high level task
Discrete Abstraction
Binary Propositions
LTL formula φ
Automaton
Hybrid Controller
Correct robot motion and action
1
2
3
4
5
6
7
8
9
10
11
12
13
Known workspace
Dynamic environment

Linear Temporal Logic (LTL)

Syntax:

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

Semantics: Truth is evaluated along infinite computation paths σ ((a,b),a,a,a… (a,b),(a,b),(a,c),(a,c),…)



$\sigma, i \models \pi$ iff $\pi \in \sigma(i)$

$\sigma, i \models \neg\varphi$ if $\sigma, i \not\models \varphi$

$\sigma, i \models \varphi_1 \vee \varphi_2$ if $\sigma, i \models \varphi_1$ or $\sigma, i \models \varphi_2$

"next" $\sigma, i \models \bigcirc\varphi$ if $\sigma, i+1 \models \varphi$

"always" $\sigma, i \models \Box\varphi$ if for all $k \geq i$ $\sigma, k \models \varphi$

"eventually" $\sigma, i \models \Diamond\varphi$ if there exists $k \geq i$ such that $\sigma, k \models \varphi$

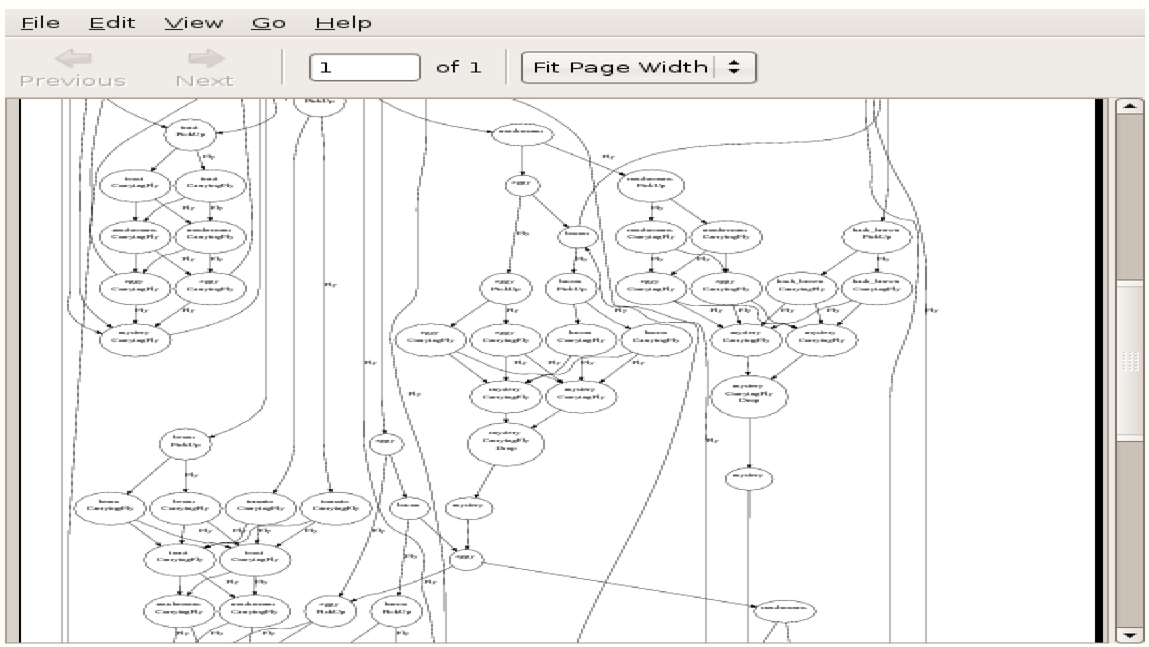
"until" $\sigma, i \models \varphi_1 \mathcal{U} \varphi_2$ if there exists $k \geq i$ such that $\sigma, k \models \varphi_2$, and for all $i \leq j < k$ we have $\sigma, j \models \varphi_1$

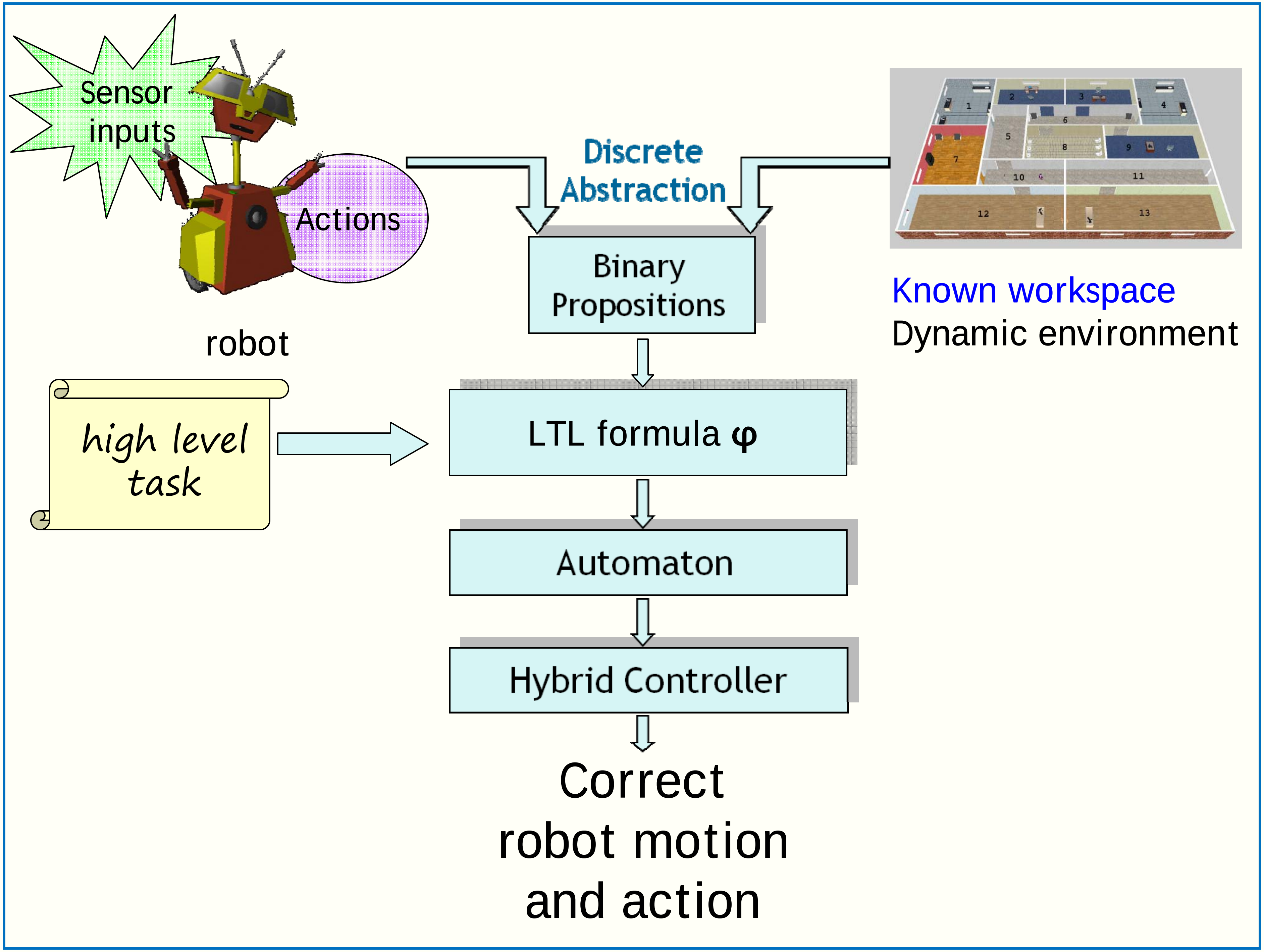
"Search rooms 1,2,3 and 4. If you see a dead body, abandon the search and go to room 11. If you see a bomb, pick it up and take it to room 13 and then resume the search."

$$
\begin{array}{l}
\vdots \\
\bigwedge \square(r_1 \rightarrow (\square\, r_1 \,\square\, \square\, r_5)) \\
\bigwedge \square(r_2 \rightarrow (\square\, r_2 \,\square\, \square\, r_6)) \\
\vdots \\
\bigwedge \square\square(sawDead \rightarrow r_{11}) \\
\bigwedge \square\square((haveBomb \wedge \neg sawDead) \rightarrow r_{13})
\end{array}
$$

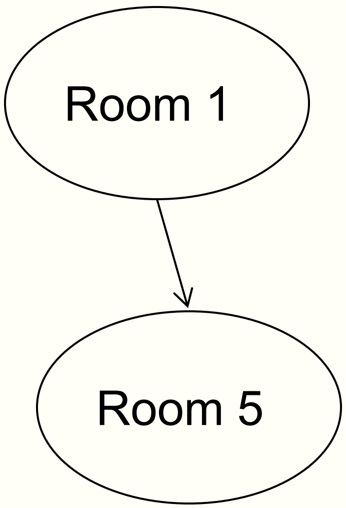
Automaton synthesis

- LTL formula converted to an automaton such that every execution is guaranteed to satisfy the formula (achieve the task) – if feasible




Sensor inputs
Actions
robot
high level task
Discrete Abstraction
Binary Propositions
LTL formula φ
Automaton
Hybrid Controller
Correct robot motion and action
Known workspace
Dynamic environment

Hybrid Controller



Bisimilar low-level controllers:
PAR or Feedback Control

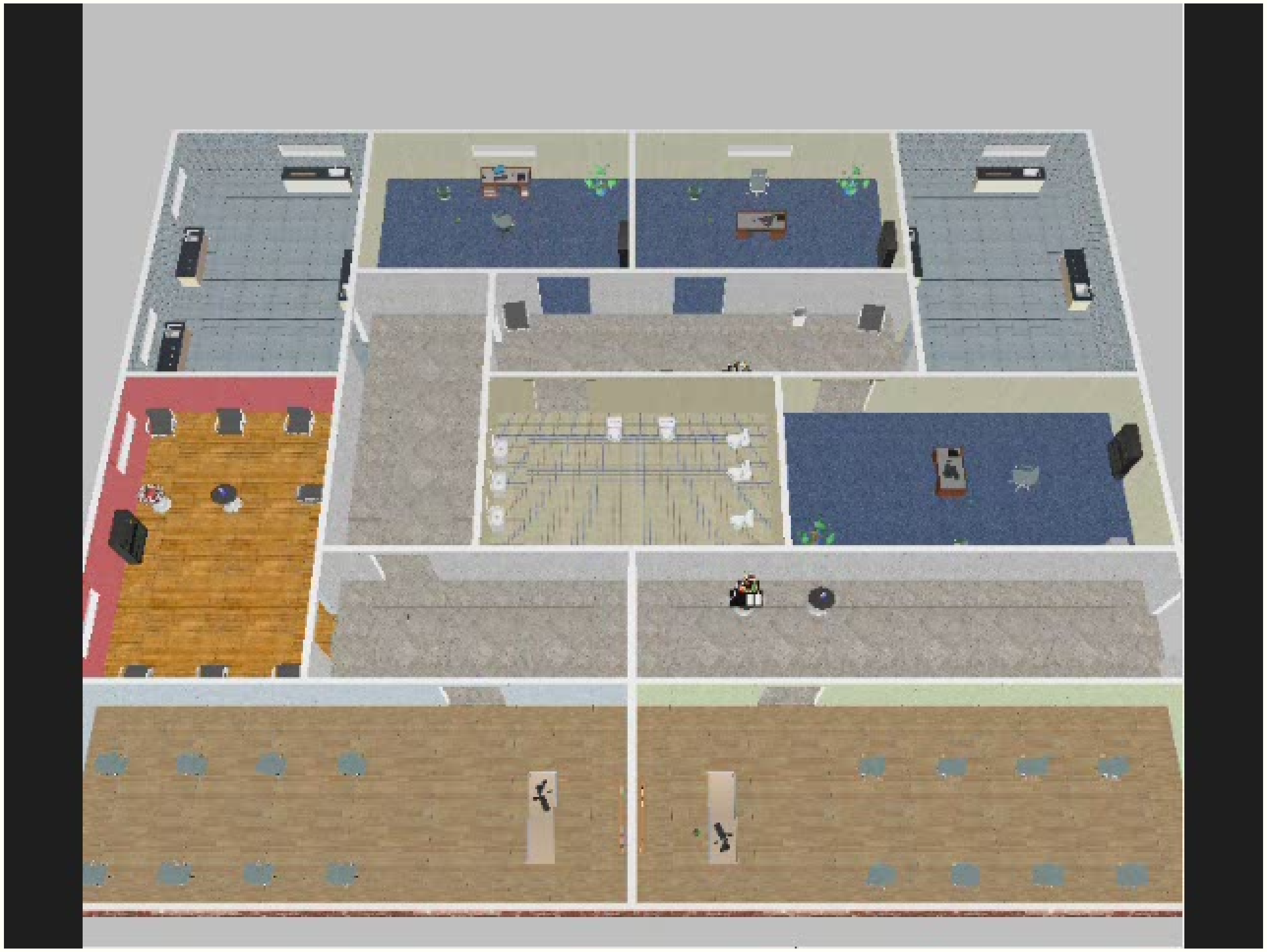
Guarantee

- If the task is feasible, a controller will be created and the robot's behavior will be correct, if the environment behaves well.

Simulation

Challenge

"If you see a bomb, pick it up and take it to room 13 and then resume the search"



Parameterized Action Representation

Ontology for simple and complex physical behaviors.

- Natural language and animation intermediary
- Applications: VET, ATOV
- Action and Object representations
- Stored in Hierarchies
- Uninstantiated and instantiated

Information in Effective Instructions

- Core action semantics (e.g. “remove”)
- Action/sub-action structure
- Participants (agent, objects)
- Path, manner, purpose information (“context”)
- Initiation conditions (applicability | preconditions)
- Termination conditions (success or failure cases)

PAR Actions

- *core semantics:* motion, force, state-change, paths
- *participants:* agent, objects
- *purpose:* state to achieve, action to generate, etc.
- *manner:* how to perform action (e.g. "carefully")
- *type:* aleatoric, reactive, opportunistic
- *duration:* timing, iteration, or extent; e.g., "for 6 seconds", "between 5 and 6 times"
- *sub-steps:* actions to perform to accomplish action (includes parallel constructs)
- *next-step:* next action to be performed
- *super-step:* parent action
- *conditions: prior, post*

Object Representation

type object representation =	
(name:	STRING;
is agent:	BOOLEAN;
properties:	sequence property-specification;
status:	status-specification;
posture:	posture-specification;
location:	object representation;
contents:	*sequence* object representation;
capabilities:	*sequence* parameterized action;
relative directions:	*sequence* relative-direction-specification;
special directions:	*sequence* special-direction-specification;
sites:	*sequence* site-type-specification;
bounding volume	bounding-volume-specification;
coordinate system	site;
position:	vector;
velocity:	vector;
acceleration:	vector;
orientation:	vector;
data:	ANY-TYPE).

World Model

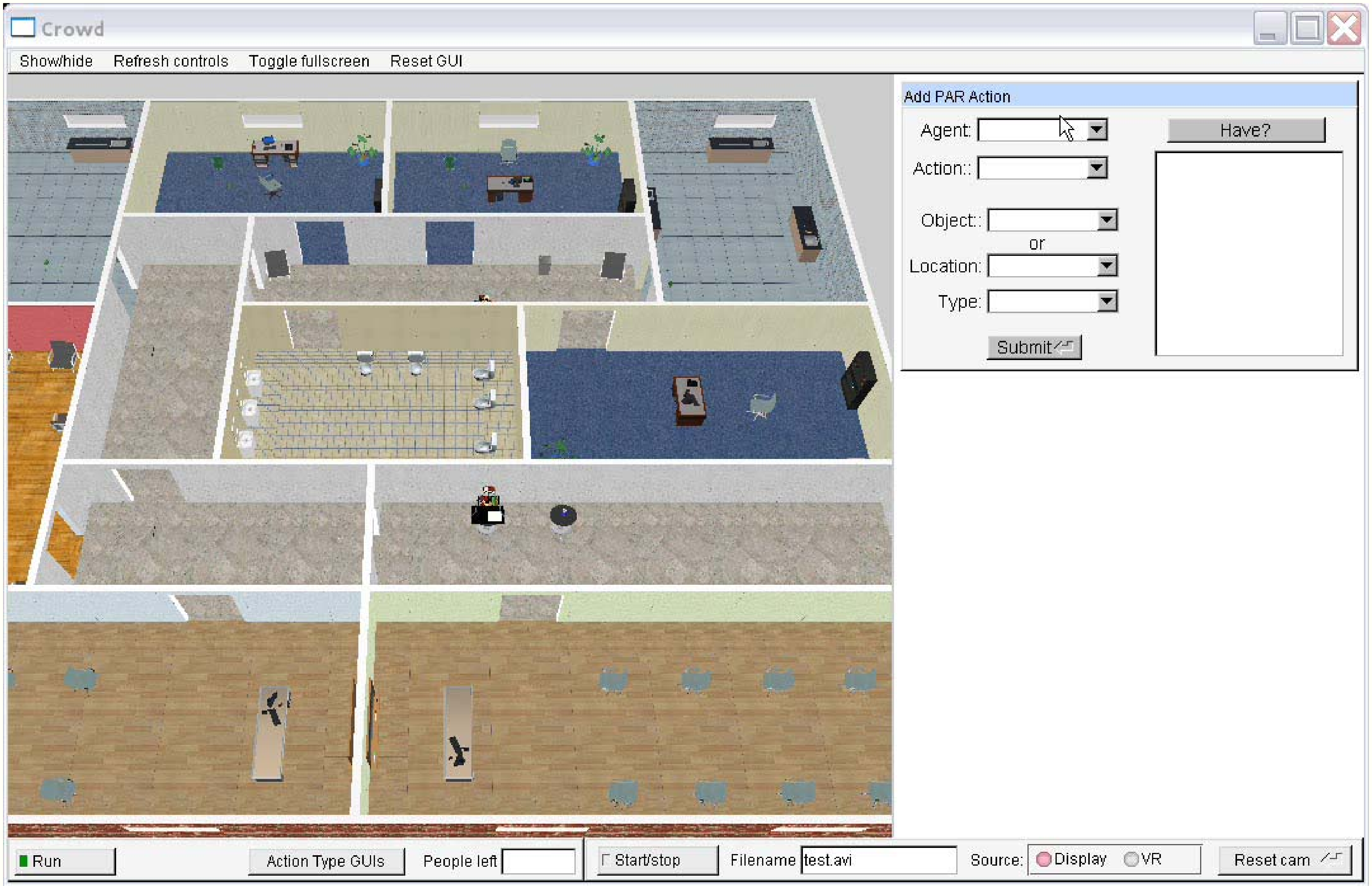
NL: Murray, pickup bomb quickly

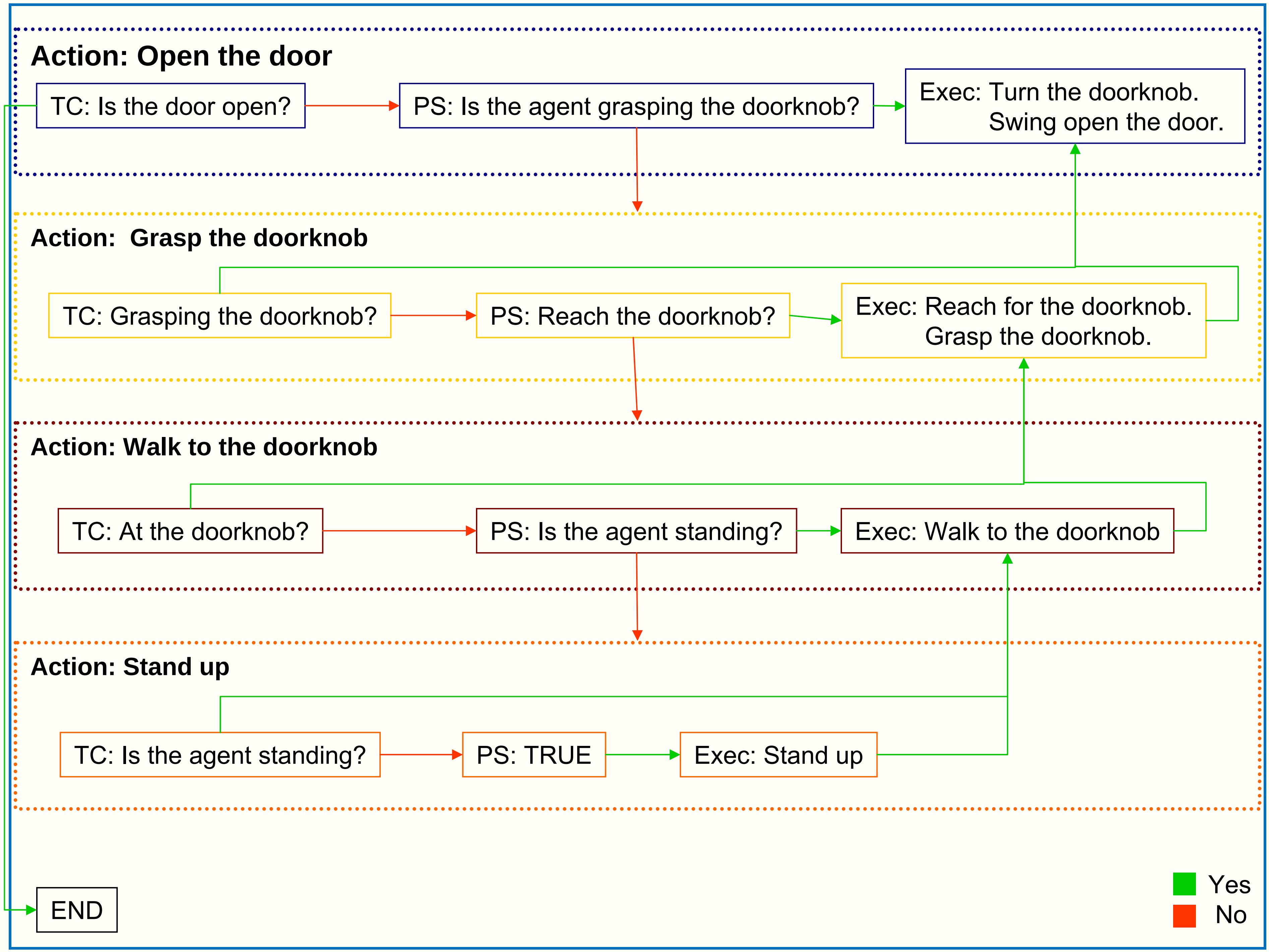
PAR: Agent: *Murray* Action: *PickUp*

Object: *Bomb* Manner: *quickly*

Animation:

Murray Interactive Demo




Action: Open the door
TC: Is the door open?
PS: Is the agent grasping the doorknob?
Exec: Turn the doorknob. Swing open the door.
Action: Grasp the doorknob
TC: Grasping the doorknob?
PS: Reach the doorknob?
Exec: Reach for the doorknob. Grasp the doorknob.
Action: Walk to the doorknob
TC: At the doorknob?
PS: Is the agent standing?
Exec: Walk to the doorknob
Action: Stand up
TC: Is the agent standing?
PS: TRUE
Exec: Stand up
END
Yes
No

PAR Summary

- Data driven
- Includes a world model
- Provides context
- Captures semantics
- Links to other software systems
- Levels of detail
- Reusable
- Composeable