

The LLM Proposes, the Robot Disposes

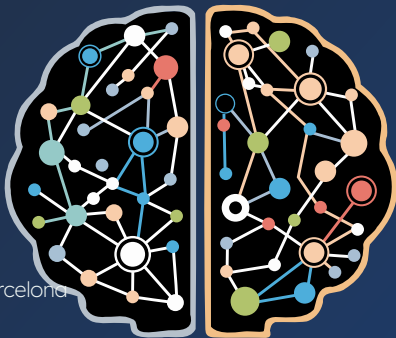
An open neuro-symbolic architecture for situated social robots

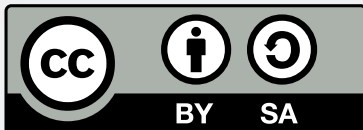
RSE 2026

S. Lemaignan¹, L. Ferrini², F. Gebellí², L. Juricic², R. Ros¹

¹ SocialMinds lab, Artificial Intelligence Research Institute IIIA-CSIC, Barcelona

² PAL Robotics, Barcelona



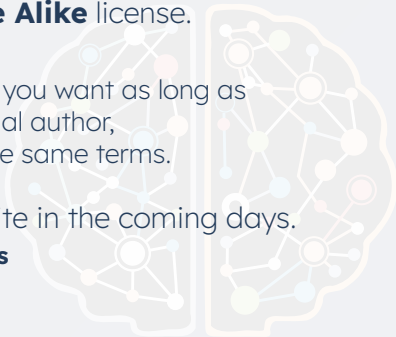


This presentation is released under the terms of the **Creative Commons Attribution-Share Alike** license.

You are free to reuse it and modify it as much as you want as long as
(1) you mention me as being the original author,
(2) you re-share your presentation under the same terms.

The slides will be made available on the lab website in the coming days.

<https://socialminds.iiia.csic.es>



PART I

Motivation and problem statement

The “bring me a coffee” scenario, used as a running example.

```
user_01 desires bring(coffee, user_01)
```



“Can you bring me a coffee?” (at PAL Robotics)

Requirements of the scenario

No entity named “coffee” is present in the room, the robot cannot operate the coffee machine, and the person in the kitchen has not been asked for help.



Requirements of the scenario

No entity named “coffee” is present in the room, the robot cannot operate the coffee machine, and the person in the kitchen has not been asked for help.

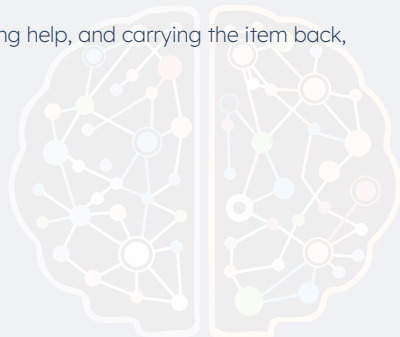
1. **Inference over the request.** “Coffee” refers to a coffee machine, located in a kitchen the robot must have mapped.



Requirements of the scenario

No entity named “coffee” is present in the room, the robot cannot operate the coffee machine, and the person in the kitchen has not been asked for help.

1. **Inference over the request.** “Coffee” refers to a coffee machine, located in a kitchen the robot must have mapped.
2. **Long-horizon planning.** Navigation, locating a person, requesting help, and carrying the item back, with steps that depend on who is present.



Requirements of the scenario

No entity named “coffee” is present in the room, the robot cannot operate the coffee machine, and the person in the kitchen has not been asked for help.

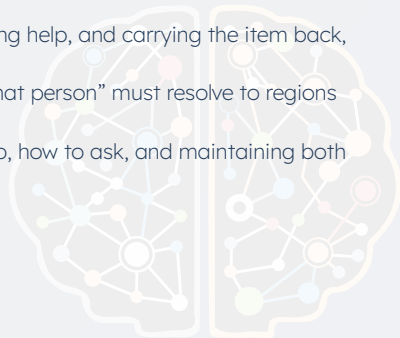
1. **Inference over the request.** “Coffee” refers to a coffee machine, located in a kitchen the robot must have mapped.
2. **Long-horizon planning.** Navigation, locating a person, requesting help, and carrying the item back, with steps that depend on who is present.
3. **Grounding of linguistic concepts.** “Kitchen”, “the table” and “that person” must resolve to regions of the environment.



Requirements of the scenario

No entity named “coffee” is present in the room, the robot cannot operate the coffee machine, and the person in the kitchen has not been asked for help.

1. **Inference over the request.** “Coffee” refers to a coffee machine, located in a kitchen the robot must have mapped.
2. **Long-horizon planning.** Navigation, locating a person, requesting help, and carrying the item back, with steps that depend on who is present.
3. **Grounding of linguistic concepts.** “Kitchen”, “the table” and “that person” must resolve to regions of the environment.
4. **Coordination through dialogue.** Reasoning about who can help, how to ask, and maintaining both conversations.

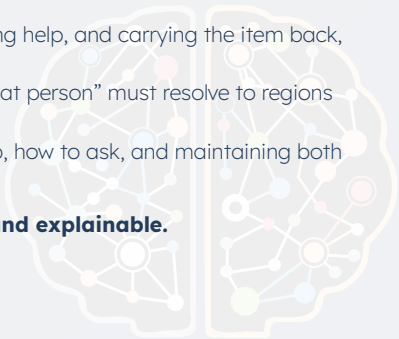


Requirements of the scenario

No entity named “coffee” is present in the room, the robot cannot operate the coffee machine, and the person in the kitchen has not been asked for help.

1. **Inference over the request.** “Coffee” refers to a coffee machine, located in a kitchen the robot must have mapped.
2. **Long-horizon planning.** Navigation, locating a person, requesting help, and carrying the item back, with steps that depend on who is present.
3. **Grounding of linguistic concepts.** “Kitchen”, “the table” and “that person” must resolve to regions of the environment.
4. **Coordination through dialogue.** Reasoning about who can help, how to ask, and maintaining both conversations.

A fifth requirement: **the decision process should remain auditable and explainable.**



Limitations of end-to-end VLA controllers

SayCan · $\pi_{0.5}$ · Gemini Robotics

VLAs perform long-horizon manipulation in unseen environments, including cleaning kitchens and making beds. For robots sharing physical and social spaces with humans over extended periods, three limitations remain.

No model of other agents

VLAs and VLMs maintain no representation of the humans they interact with: their beliefs, engagement, intentions or perspectives.



Limitations of end-to-end VLA controllers

SayCan · $\pi_{0.5}$ · Gemini Robotics

VLAs perform long-horizon manipulation in unseen environments, including cleaning kitchens and making beds. For robots sharing physical and social spaces with humans over extended periods, three limitations remain.

No model of other agents

VLAs and VLMs maintain no representation of the humans they interact with: their beliefs, engagement, intentions or perspectives.

No structured world model

Spatial knowledge is encoded implicitly in the weights. There is no explicit, queryable model to share across subsystems or to populate from non-visual sources.



Limitations of end-to-end VLA controllers

SayCan · $\pi_{0.5}$ · Gemini Robotics

VLAs perform long-horizon manipulation in unseen environments, including cleaning kitchens and making beds. For robots sharing physical and social spaces with humans over extended periods, three limitations remain.

No model of other agents

VLAs and VLMs maintain no representation of the humans they interact with: their beliefs, engagement, intentions or perspectives.

No structured world model

Spatial knowledge is encoded implicitly in the weights. There is no explicit, queryable model to share across subsystems or to populate from non-visual sources.

Limited auditability

It is difficult to inspect why an action was selected, to verify compliance with a safety constraint, or to explain the behaviour to a user or a regulator.



Limitations of end-to-end VLA controllers

SayCan · $\pi_{0.5}$ · Gemini Robotics

VLAs perform long-horizon manipulation in unseen environments, including cleaning kitchens and making beds. For robots sharing physical and social spaces with humans over extended periods, three limitations remain.

No model of other agents

VLAs and VLMs maintain no representation of the humans they interact with: their beliefs, engagement, intentions or perspectives.

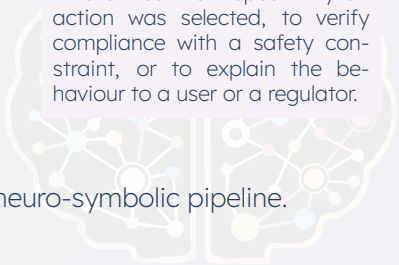
No structured world model

Spatial knowledge is encoded implicitly in the weights. There is no explicit, queryable model to share across subsystems or to populate from non-visual sources.

Limited auditability

It is difficult to inspect why an action was selected, to verify compliance with a safety constraint, or to explain the behaviour to a user or a regulator.

We position the LLM as one component within a broader neuro-symbolic pipeline.



Three challenges addressed by the architecture

Challenge	Difficulty for an LLM-driven system	Component
-----------	-------------------------------------	-----------



Three challenges addressed by the architecture

Challenge	Difficulty for an LLM-driven system	Component
Socio-physical grounding	Language models are disembodied; the robot's 3D perception is not available to them.	ROS4HRI , reMap and KnowledgeCore : HRI + voxel-level facts lifted into RDF triples, serialised into the prompt.



Three challenges addressed by the architecture

Challenge	Difficulty for an LLM-driven system	Component
Socio-physical grounding	Language models are disembodied; the robot's 3D perception is not available to them.	ROS4HRI , reMap and KnowledgeCore : HRI + voxel-level facts lifted into RDF triples, serialised into the prompt.
Locus of control	An LLM action is text generation. A robot action takes time, has physical side effects, and can be unsafe.	Intents and mission controller : input modules propose, the controller validates and dispatches.



Three challenges addressed by the architecture

Challenge	Difficulty for an LLM-driven system	Component
Socio-physical grounding	Language models are disembodied; the robot's 3D perception is not available to them.	ROS4HRI , reMap and KnowledgeCore : HRI + voxel-level facts lifted into RDF triples, serialised into the prompt.
Locus of control	An LLM action is text generation. A robot action takes time, has physical side effects, and can be unsafe.	Intents and mission controller : input modules propose, the controller validates and dispatches.
Multi-party, multi-turn dialogue	Chat interfaces assume a single user, a single thread, and no floor management.	Per-person and per-group contexts with priority-based arbitration.

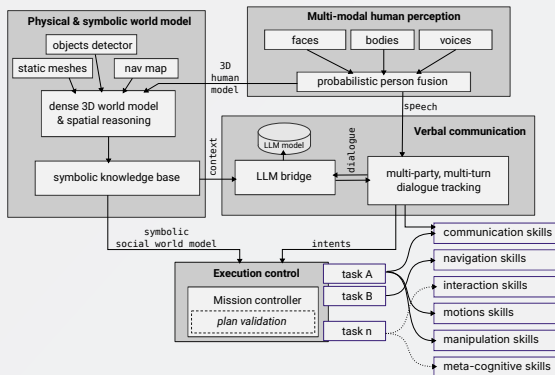
PART II

Architecture principles

Shared state is mediated by the symbolic knowledge base rather than by ad-hoc inter-component APIs.

```
architecture integratesWith ROS 2
```

Bidirectional, event-driven and deliberative



Data flow

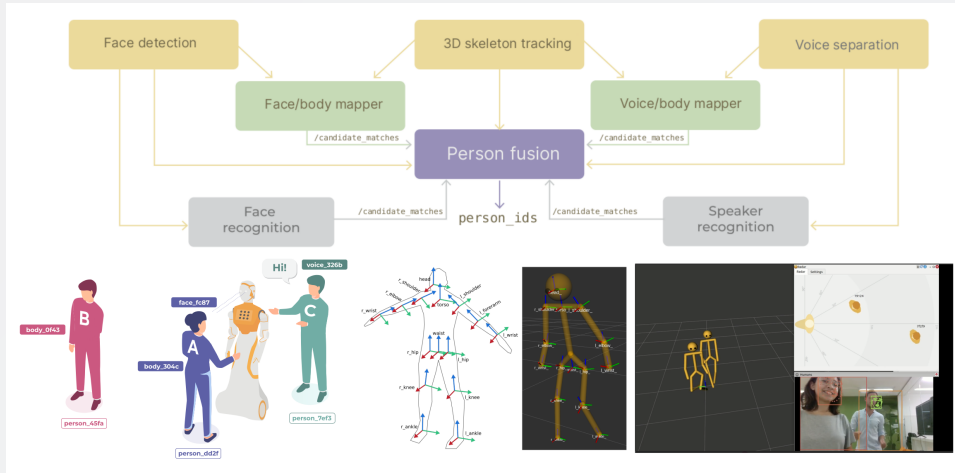
- Perception nodes publish detections of people, objects and utterances continuously. reMap and the person fusion pipeline assert them as **RDF triples**.
- Utterances are paired with a symbolic **environment context** assembled from the knowledge base and forwarded to the LLM-based chatbot.
- The chatbot emits **intents**. The mission controller validates the proposed plan and dispatches tasks and skills.
- Spoken output, facial expression, gaze targets, LED patterns and gestures are synchronised through the dialogue manager and the attention manager.

Socio-physical grounding

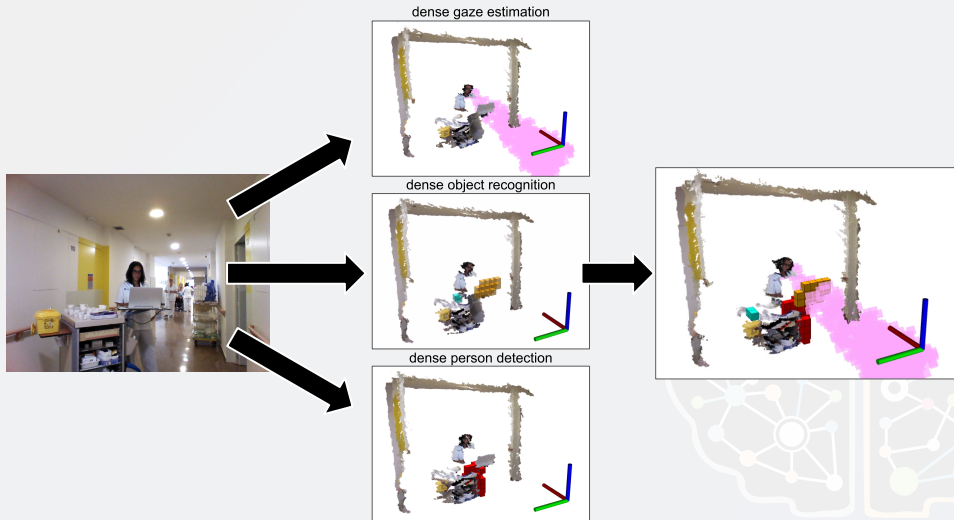
Anchoring linguistic concepts in the robot's situated perception of the physical world and the social context.

cup_456 isOn table_01

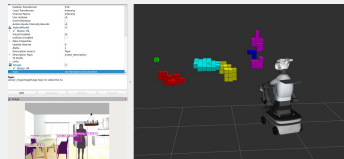
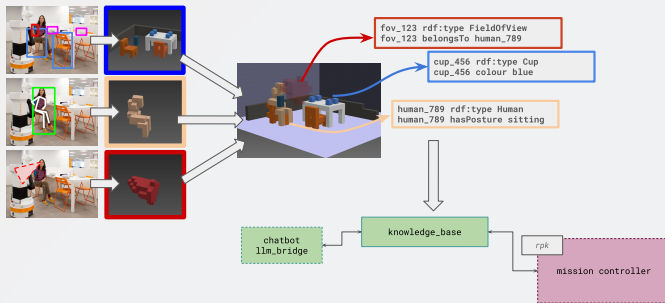
ROS4HRI: s a standard for HRI in ROS



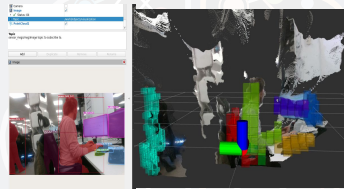
reMap: continuous grounding, in both directions



reMap: continuous grounding, in both directions



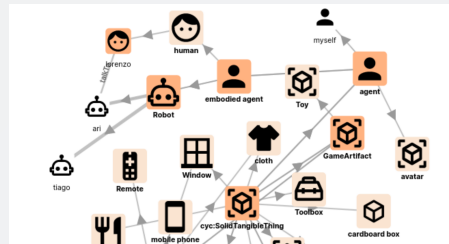
In simulation...



...or in the real-world

KnowledgeCore: a knowledge base with human-level semantics

RDF triples under an OWL ontology, **rdflib** plus the **reasonable** OWL2-RL reasoner, running as a ROS 2 lifecycle node.

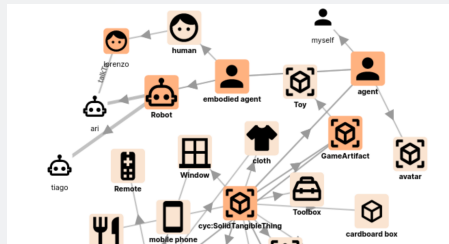


kb_explorer, the interactive browser of KnowledgeCore

KnowledgeCore: a knowledge base with human-level semantics

RDF triples under an OWL ontology, **rdflib** plus the **reasonable** OWL2-RL reasoner, running as a ROS 2 lifecycle node.

- o **Multiple knowledge models.** Separate beliefs per agent, providing the foundation for perspective taking and theory of mind. Not exploited in the current deployments.

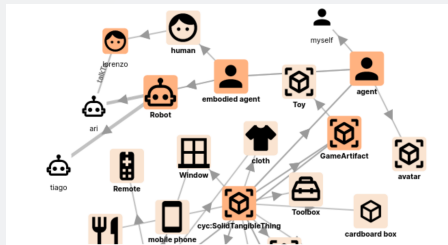


kb_explorer, the interactive browser of KnowledgeCore

KnowledgeCore: a knowledge base with human-level semantics

RDF triples under an OWL ontology, **rdflib** plus the **reasonable** OWL2-RL reasoner, running as a ROS 2 lifecycle node.

- o **Multiple knowledge models.** Separate beliefs per agent, providing the foundation for perspective taking and theory of mind. Not exploited in the current deployments.
- o **Event system.** Components subscribe to knowledge patterns (a new **Human** instance is added, **?agent desires ?action** becomes true) and are notified asynchronously rather than polling.

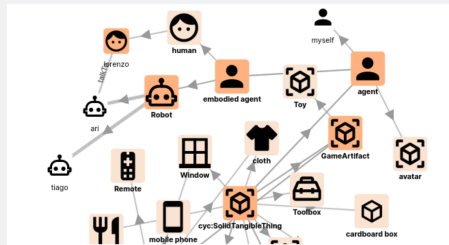


kb_explorer, the interactive browser of KnowledgeCore

KnowledgeCore: a knowledge base with human-level semantics

RDF triples under an OWL ontology, **rdflib** plus the **reasonable** OWL2-RL reasoner, running as a ROS 2 lifecycle node.

- o **Multiple knowledge models.** Separate beliefs per agent, providing the foundation for perspective taking and theory of mind. Not exploited in the current deployments.
- o **Event system.** Components subscribe to knowledge patterns (a new **Human** instance is added, **?agent desires ?action** becomes true) and are notified asynchronously rather than polling.
- o **Transient knowledge.** Facts carry lifespans and are automatically retracted, giving three memory profiles: short-term (seconds), episodic (minutes) and long-term.



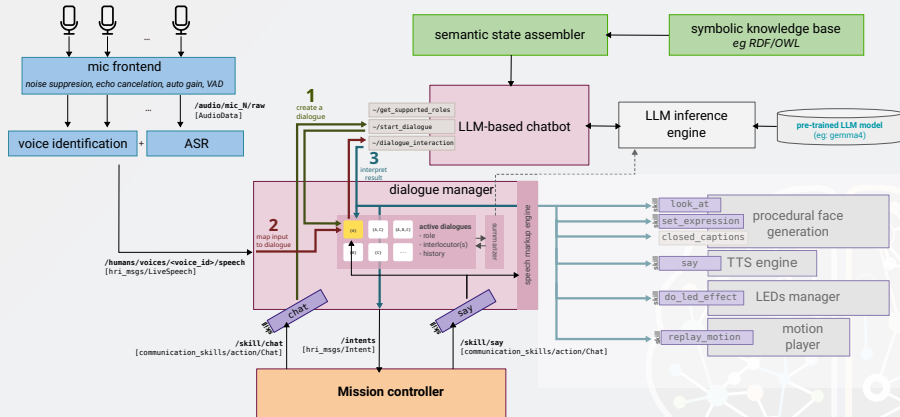
kb_explorer, the interactive browser of KnowledgeCore

Multi-party dialogue management

Verbal communication on a robot is multi-modal, multi-party, and continuously re-grounded in the world model.

dialogue_7	hasInterlocutor	group_02
------------	-----------------	----------

The verbal communication pipeline



The dialogue manager coordinates three communication skills (chat, ask, say), manages concurrent dialogues with priority handling, and interfaces with the chatbot engine, the semantic state assembler, and the multi-modal output channels.

Prompting strategy

Initial system prompt

> Used when a new conversation is started

You are a friendly robot ... You are always helpful ...
Your aim is to extract the user intent from their request...
Your response must be a JSON object...

Propose a plan of skills to achieve the detected goal. The plan must be a list of skills ...

Here is the list of skills... say, navigate_to ...

Environment system prompt

> Appended to the history if there are environment updates

These are the available rooms: kitchen, living room, bedroom

You are currently in the living room

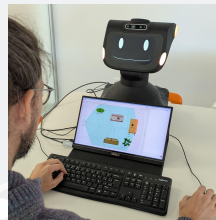
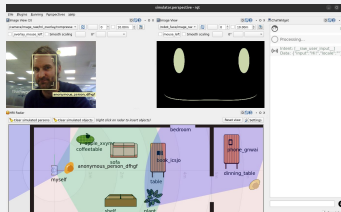
These are the objects that you currently see:

- The book_icsjo, which is on the table has color blue, and has name Pride_and_prejudice.
- ...

User prompt

> Coming from automatic speech recognition or chat widget

Can you bring me a coffee?



Generated Response from LLM

```
ASSISTANT: {
  "say": "Sure, I'll get your coffee from the kitchen.",
  "user_intent": {
    "type": "__intent_start_activity__",
    "object": "bring_coffee_from_kitchen",
    "plan": [
      {
        "skill": "navigate_to",
        "params": {
          "goal": "kitchen",
          "result": null
        },
        "skill": "look_for_human",
        "params": {
          "patterns": [],
          "result": null
        },
        "skill": "ask_human_for_help",
        "params": {
          "text": "Can you bring me a coffee from the kitchen?",
          "recipient": "?look_for_human.found_entities",
          "result": null
        },
        "skill": "navigate_to",
        "params": {
          "goal": "living_room",
          "result": null
        },
        "skill": "say",
        "params": {
          "input": "Here is your coffee.",
          "result": null
        }
      ]
    ]
  }
}
```

Locus of control

An action for a text-based LLM is generating a response. A robot action takes time, can be unsafe, and can cause interaction breakdown.

input modules **propose** the mission controller disposes

Intent representation

`hri_actions_msgs/Intent` on `/intents`

- o **name**: one of the pre-defined intents (ENGAGE_WITH, MOVE_TO, GUIDE, BRING_OBJECT, SAY, START_ACTIVITY) or a user-defined name.

subsystems publishing on `/intents`

LLM chatbot	publishes	intent from an utterance
touchscreen	publishes	intent from a button press
operator console	publishes	remote intent
battery monitor	publishes	GO_TO(dock), high priority

The mission controller aggregates these messages, arbitrates using the priority field, validates the proposed action against the current world state, and dispatches it. Recognition of what the user wants is decoupled from the decision of whether and how to act on it.

Intent representation

`hri_actions_msgs/Intent` on `/intents`

- o **name:** one of the pre-defined intents (`ENGAGE_WITH`, `MOVE_TO`, `GUIDE`, `BRING_OBJECT`, `SAY`, `START_ACTIVITY`) or a user-defined name.
- o **data:** a JSON payload parametrising the action through linguistic thematic roles (agent, object, goal, recipient), kept close to the predicate structure of the OpenRobots ontology.

subsystems publishing on `/intents`

LLM chatbot	publishes	intent from an utterance
touchscreen	publishes	intent from a button press
operator console	publishes	remote intent
battery monitor	publishes	GO_TO(dock), high priority

The mission controller aggregates these messages, arbitrates using the priority field, validates the proposed action against the current world state, and dispatches it. Recognition of what the user wants is decoupled from the decision of whether and how to act on it.

Intent representation

`hri_actions_msgs/Intent` on `/intents`

- o **name:** one of the pre-defined intents (`ENGAGE_WITH`, `MOVE_TO`, `GUIDE`, `BRING_OBJECT`, `SAY`, `START_ACTIVITY`) or a user-defined name.
- o **data:** a JSON payload parametrising the action through linguistic thematic roles (agent, object, goal, recipient), kept close to the predicate structure of the OpenRobots ontology.
- o **source:** a ROS4HRI person id, the robot itself via `__myself__`, or a remote supervisor.

subsystems publishing on `/intents`

LLM chatbot	publishes	intent from an utterance
touchscreen	publishes	intent from a button press
operator console	publishes	remote intent
battery monitor	publishes	GO_TO(dock), high priority

The mission controller aggregates these messages, arbitrates using the priority field, validates the proposed action against the current world state, and dispatches it. Recognition of what the user wants is decoupled from the decision of whether and how to act on it.

Intent representation

`hri_actions_msgs/Intent` on `/intents`

- o **name**: one of the pre-defined intents (ENGAGE_WITH, MOVE_TO, GUIDE, BRING_OBJECT, SAY, START_ACTIVITY) or a user-defined name.
- o **data**: a JSON payload parametrising the action through linguistic thematic roles (agent, object, goal, recipient), kept close to the predicate structure of the OpenRobots ontology.
- o **source**: a ROS4HRI person id, the robot itself via `__myself__`, or a remote supervisor.
- o **modality**: speech, motion, touchscreen, internal or other. Priority and confidence are optional.

subsystems publishing on `/intents`

LLM chatbot	publishes	intent from an utterance
touchscreen	publishes	intent from a button press
operator console	publishes	remote intent
battery monitor	publishes	GO_TO(dock), high priority

The mission controller aggregates these messages, arbitrates using the priority field, validates the proposed action against the current world state, and dispatches it. Recognition of what the user wants is decoupled from the decision of whether and how to act on it.

Intent representation

`hri_actions_msgs/Intent` on `/intents`

- o **name**: one of the pre-defined intents (ENGAGE_WITH, MOVE_TO, GUIDE, BRING_OBJECT, SAY, START_ACTIVITY) or a user-defined name.
- o **data**: a JSON payload parametrising the action through linguistic thematic roles (agent, object, goal, recipient), kept close to the predicate structure of the OpenRobots ontology.
- o **source**: a ROS4HRI person id, the robot itself via `__myself__`, or a remote supervisor.
- o **modality**: speech, motion, touchscreen, internal or other. Priority and confidence are optional.

Intents are reserved for requests with side effects on the robot's physical or social state. Purely informational exchanges, such as answering a weather query, are handled by the dialogue manager without entering the intent pipeline.

subsystems publishing on `/intents`

LLM chatbot	publishes	intent from an utterance
touchscreen	publishes	intent from a button press
operator console	publishes	remote intent
battery monitor	publishes	GO_TO(dock), high priority

The mission controller aggregates these messages, arbitrates using the priority field, validates the proposed action against the current world state, and dispatches it. Recognition of what the user wants is decoupled from the decision of whether and how to act on it.

Mission, task, skill

Action model, following RobMoSys design principles

Mission

“Home assistant” or “drug delivery”. The mission reacts to incoming intents and starts tasks. The mission controller orchestrates the robot’s behaviour.

Task

Fetching a book, greeting a person. A task combines multiple skills, each sub-goal requiring several of them.

Skill

navigate, say, grasp,
ask_human_for_help.
Parametrised and named at a granularity meaningful to a human designer or a task planner.



Mission, task, skill

Action model, following RobMoSys design principles

Mission

“Home assistant” or “drug delivery”. The mission reacts to incoming intents and starts tasks. The mission controller orchestrates the robot’s behaviour.

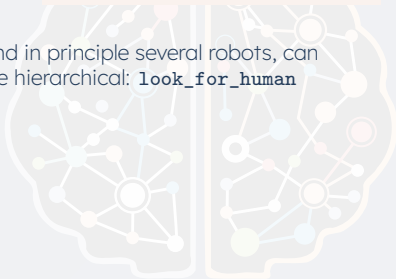
Task

Fetching a book, greeting a person. A task combines multiple skills, each sub-goal requiring several of them.

Skill

`navigate`, `say`, `grasp`,
`ask_human_for_help`.
Parametrised and named at a granularity meaningful to a human designer or a task planner.

- **Definition separate from implementation.** Several modules, and in principle several robots, can provide competing implementations of the same skill. Skills may be hierarchical: `look_for_human` invokes `look_at`.



Mission, task, skill

Action model, following RobMoSys design principles

Mission

“Home assistant” or “drug delivery”. The mission reacts to incoming intents and starts tasks. The mission controller orchestrates the robot’s behaviour.

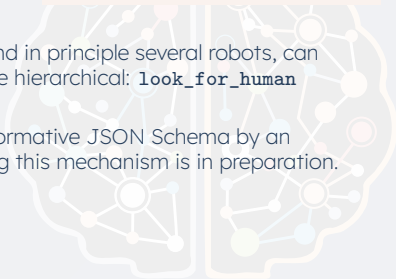
Task

Fetching a book, greeting a person. A task combines multiple skills, each sub-goal requiring several of them.

Skill

`navigate`, `say`, `grasp`,
`ask_human_for_help`.
Parametrised and named at a granularity meaningful to a human designer or a task planner.

- **Definition separate from implementation.** Several modules, and in principle several robots, can provide competing implementations of the same skill. Skills may be hierarchical: `look_for_human` invokes `look_at`.
- **Manifests are embedded in `package.xml`**, validated against a normative JSON Schema by an `ament_lint` extension in CI. A ROS Enhancement Proposal covering this mechanism is in preparation.



Mission, task, skill

Action model, following RobMoSys design principles

Mission

“Home assistant” or “drug delivery”. The mission reacts to incoming intents and starts tasks. The mission controller orchestrates the robot’s behaviour.

Task

Fetching a book, greeting a person. A task combines multiple skills, each sub-goal requiring several of them.

Skill

`navigate`, `say`, `grasp`,
`ask_human_for_help`.
Parametrised and named at a granularity meaningful to a human designer or a task planner.

- **Definition separate from implementation.** Several modules, and in principle several robots, can provide competing implementations of the same skill. Skills may be hierarchical: `look_for_human` invokes `look_at`.
- **Manifests are embedded in `package.xml`**, validated against a normative JSON Schema by an `ament_lint` extension in CI. A ROS Enhancement Proposal covering this mechanism is in preparation.
- **Uniform metadata, result and feedback structures** on every skill, so the mission controller acts as a generic executor through a single code path.

Validation before dispatch

Mission controller · plan checker [WIP!]

1 · Skill availability

The current capabilities of the robot are checked against the skills specified in the plan, preventing execution of actions that cannot be supported.



Validation before dispatch

Mission controller · plan checker [WIP!]

1 · Skill availability

The current capabilities of the robot are checked against the skills specified in the plan, preventing execution of actions that cannot be supported.

2 · Parameter completeness

Dependencies between skills are identified and all required inputs must be computable at runtime, including outputs propagated from preceding skills.



Validation before dispatch

Mission controller · plan checker [WIP!]

1 · Skill availability

The current capabilities of the robot are checked against the skills specified in the plan, preventing execution of actions that cannot be supported.

2 · Parameter completeness

Dependencies between skills are identified and all required inputs must be computable at runtime, including outputs propagated from preceding skills.

A higher-priority intent pauses the current execution. Execution failures lead to re-planning or to aborting the goal, and the user is informed of the deviation.



Validation before dispatch

Mission controller · plan checker [WIP!]

1 · Skill availability

The current capabilities of the robot are checked against the skills specified in the plan, preventing execution of actions that cannot be supported.

2 · Parameter completeness

Dependencies between skills are identified and all required inputs must be computable at runtime, including outputs propagated from preceding skills.

A higher-priority intent pauses the current execution. Execution failures lead to re-planning or to aborting the goal, and the user is informed of the deviation.

input modules **propose** the mission controller disposes



PART VI

Real-world deployments

A home assistant scenario implemented in the lab, and an office delivery robot operated autonomously for one month.

home_assistant and office_delivery

“Bring me a coffee” lab deployment



Office internal delivery system

DIEGo · Deliver Internal Employee Goods



DIEGo operating in the open-plan office.

1 month of autonomous operation
~**100** employees in the open-plan office
Slack user interface for all coordination
2 tasks delivery and recovery



Three claims

Summary

Claim 1: grounding through a shared symbolic layer

A queryable RDF world model between the 3D perception and the LLM lets each side operate in its own representation, and gives every other subsystem access to the same state.



Three claims

Summary

Claim 1: grounding through a shared symbolic layer

A queryable RDF world model between the 3D perception and the LLM lets each side operate in its own representation, and gives every other subsystem access to the same state.

Claim 2: explicit arbitration of the locus of control

Every request, whether spoken, entered on a touchscreen, issued remotely or self-generated, is routed through a single intent message, and plans are validated before dispatch. We are aware of no prior published neuro-symbolic social-robot architecture that treats this aspect explicitly.



Three claims

Summary

Claim 1: grounding through a shared symbolic layer

A queryable RDF world model between the 3D perception and the LLM lets each side operate in its own representation, and gives every other subsystem access to the same state.

Claim 2: explicit arbitration of the locus of control

Every request, whether spoken, entered on a touchscreen, issued remotely or self-generated, is routed through a single intent message, and plans are validated before dispatch. We are aware of no prior published neuro-symbolic social-robot architecture that treats this aspect explicitly.

Claim 3: inspectable intermediate representations

Knowledge base contents, assembled prompts, emitted intents and validated plans are logged, replayable and visualisable, which supports explanation, audit and regulatory compliance.



Three claims

Summary

Claim 1: grounding through a shared symbolic layer

A queryable RDF world model between the 3D perception and the LLM lets each side operate in its own representation, and gives every other subsystem access to the same state.

Claim 2: explicit arbitration of the locus of control

Every request, whether spoken, entered on a touchscreen, issued remotely or self-generated, is routed through a single intent message, and plans are validated before dispatch. We are aware of no prior published neuro-symbolic social-robot architecture that treats this aspect explicitly.

Claim 3: inspectable intermediate representations

Knowledge base contents, assembled prompts, emitted intents and validated plans are logged, replayable and visualisable, which supports explanation, audit and regulatory compliance.

The architecture is ROS 2 native, modular and open source. The LLM is a replaceable component behind a thin chatbot interface.

Featured software

socialminds.doc.iiaa.csic.es/apt/software.html

The screenshot displays a web page for ROS 2 packages. It is divided into two main sections: 'Interactions' and 'Knowledge representation'. The 'Interactions' section contains three cards: 'DM' (dialogue-manager), 'expressive-face', and 'human-description'. Each card shows a brief description, overview, main features, and installation instructions. The 'Knowledge representation' section contains three cards: 'knowledge-core', 'oro', and 'remap'. Each card shows a brief description and installation instructions.

Interactions

- DM**
dialogue-manager
Multi-party, multi-turn dialogue management with LLM integration
A ROS2 lifecycle node that handles multi-modal communication between the robot and humans.
The Dialogue Manager:
 - Tracks every conversation the robot is engaged...[More...](#)
typical: 2.2.1 all jazzy: 2.2.1 all
[Install](#)
[Source](#)
- expressive-face**
Fast, shader-based expressive robot faces with a live face editor
Overview
expressive_faces is a procedural face generator and interactive face editor for robots.
Main features
 - fully integrated with ROS 2...[More...](#)
typical: 8.7.3 amd64 jazzy: 8.8.8 amd64 + arm64
[Install](#)
[Source](#)
- human-description**
URDF-based parametric kinematic model of humans
Parameters kinematic model of humans, in URDF format, for usage in robotics and HRI
Part of the [ROS4HRI](#) project.
Install the package, and launch: `roslaunch human_description show.launch.py`
[More...](#)
typical: 2.0.2 amd64 jazzy: 2.0.2 amd64 + arm64 humble: 2.0.2 amd64
[Install](#)
[Source](#)

Knowledge representation

- knowledge-core**
ROS2 lifecycle node that handles multi-modal communication between the robot and humans.
- oro**
Fast, shader-based expressive robot faces with a live face editor
- remap**
Fast, shader-based expressive robot faces with a live face editor

The components of the architecture are released as ROS 2 packages on the lab software page, grouped by function.

Each entry lists the ROS 2 distributions it is built for, apt installation instructions, and a link to the sources.

dialogue-manager
expressive-face
human-description
knowledge-core
oro
remap

RSE 2026

Software and contact

FEATURED SOFTWARE, WITH INSTALLATION INSTRUCTIONS

socialminds.doc.iiaa.csic.es/apt/software.html

KNOWLEDGECORE

[gitlab.iiaa.csic.es/socialminds/
neurosymbolic-ai/knowledge_core](https://gitlab.iiaa.csic.es/socialminds/neurosymbolic-ai/knowledge_core)

SKILLS REP, IN PREPARATION

github.com/ros4hri/REP-skills

SKILLS AND INTENTS REFERENCE

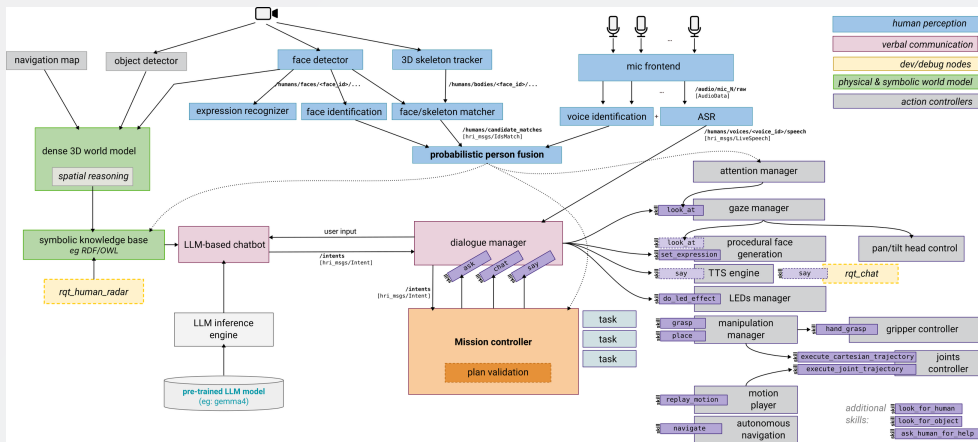
ros4hri.github.io/skills
ros4hri.github.io/reference/intents.html

CONTACT

severin.lemaignan@iiaa.csic.es

Funded by the European Union Horizon Europe programme under grant agreements no. 101070254 (CoreSense), no. 101135784 (ARISE), SAFELY and DIEGO. Social perception builds on ROS4HRI (REP-155) and on the deployment experience of the SPRING architecture.

Appendix: the implemented architecture, node by node



Blue: ROS4HRI perception · green: world model and KB · violet: verbal communication · orange: action controllers and skills · yellow: development and debugging tools.

Appendix: full trace of the example situation

Each world change, the resulting knowledge-base delta, the prompt updates it produces, and the skill executing at that moment.

- **S**: a system prompt update from the semantic state assembler
- **U**: a user prompt update, transcribed by ASR
- **A**: assistant text generated by the LLM

For brevity, the plan syntax in the LLM prompt has been modified.

reMap - knowledge base			LLM prompt	execution
kitchen	living room	corridor	+ Kitchen ref-type Zone + living-room ref-type Zone + corridor ref-type Zone	S: You are [...]. These are the available rooms: kitchen, living-room, corridor.
			+ myself isIn kitchen + bottle_123 ref-type Bottle + bottle_123 isIn kitchen	S: myself isIn kitchen, sees bottle_123, bottle_123 isIn kitchen. (exploring the environment)
			- myself isIn kitchen - myself sees bottle_123	S: myself no longer isIn kitchen, no longer sees bottle_123. (exploring the environment)
			+ myself isIn living-room + book_123 ref-type Book + myself sees book_123 + book_34 isIn living-room	S: myself isIn living-room, sees book_123, book_123 isIn kitchen. (exploring the environment)
			- myself isIn living-room - myself sees book_123	S: myself no longer isIn living-room, no longer sees book_123. (exploring the environment)
			+ myself isIn corridor + human_123 ref-type Human + myself sees human_123	S: myself isIn corridor, sees human_123. (exploring the environment)
				U: Hey, can you bring me a bottle?
				A: [say: sure!, intent: {plan: navigate(kitchen), ask human for help (can you put a bottle on my tray?), navigate(corridor), say (here it is)}]
			- myself isIn living-room - myself sees human_123	S: myself no longer isIn living-room, no longer sees book_123. (executing the navigation skill)
			+ myself isIn kitchen + myself sees bottle_456 + human_456 ref-type Human + myself sees human_456	S: myself isIn kitchen, sees bottle_123. (executing the ask human for help skill) A: [say: Can you put a bottle on my tray?, intent: {}] U: Sure!
			- myself isIn kitchen - myself sees bottle_456 - myself sees human_456	S: myself no longer isIn kitchen, no longer sees bottle_456, no longer sees human_456. (executing the navigation skill)
			+ myself isIn corridor + myself sees human_789	S: myself isIn corridor, sees human_789. A: [say: Here you are, intent: {}] (executing the say skill)

Appendix: Transparency and explainability

HEXAR · meta-cognitive skills

Meta-cognitive skills monitor other skills, plan creation and execution. They are triggered by the mission controller or by user queries, and reuse the internal logs and interfaces the architecture already exposes.

Stage 1: explainer selector

Builds a context from current events derived from internal application signals and logs, and checks the last plan for failures or invalidity. If one is found, the corresponding component explainer is selected automatically.



Appendix: Transparency and explainability

HEXAR · meta-cognitive skills

Meta-cognitive skills monitor other skills, plan creation and execution. They are triggered by the mission controller or by user queries, and reuse the internal logs and interfaces the architecture already exposes.

Stage 1: explainer selector

Builds a context from current events derived from internal application signals and logs, and checks the last plan for failures or invalidity. If one is found, the corresponding component explainer is selected automatically.

Stage 2: component explainers

One per skill, plus one for the task planner. Each monitors particular internal event signals and may apply any technical method: template-based, LLM-driven, feature-based or causal-model-based.



Appendix: Transparency and explainability

HEXAR · meta-cognitive skills

Meta-cognitive skills monitor other skills, plan creation and execution. They are triggered by the mission controller or by user queries, and reuse the internal logs and interfaces the architecture already exposes.

Stage 1: explainer selector

Builds a context from current events derived from internal application signals and logs, and checks the last plan for failures or invalidity. If one is found, the corresponding component explainer is selected automatically.

Stage 2: component explainers

One per skill, plus one for the task planner. Each monitors particular internal event signals and may apply any technical method: template-based, LLM-driven, feature-based or causal-model-based.

Routing of open user questions

In the absence of a failure, an LLM routes the user's query to the most appropriate component explainer. The design supports single-explainer selection without aggregation.



Appendix: Transparency and explainability

HEXAR · meta-cognitive skills

Meta-cognitive skills monitor other skills, plan creation and execution. They are triggered by the mission controller or by user queries, and reuse the internal logs and interfaces the architecture already exposes.

Stage 1: explainer selector

Builds a context from current events derived from internal application signals and logs, and checks the last plan for failures or invalidity. If one is found, the corresponding component explainer is selected automatically.

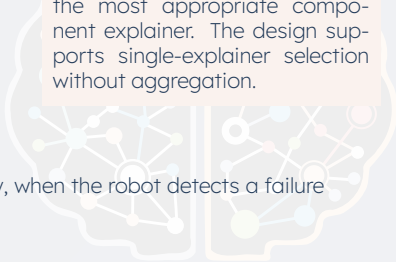
Stage 2: component explainers

One per skill, plus one for the task planner. Each monitors particular internal event signals and may apply any technical method: template-based, LLM-driven, feature-based or causal-model-based.

Routing of open user questions

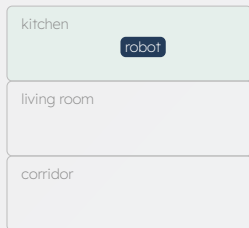
In the absence of a failure, an LLM routes the user's query to the most appropriate component explainer. The design supports single-explainer selection without aggregation.

In the office deployment, explanations are generally provided proactively, when the robot detects a failure during execution.



Trace of a single request: exploration and request

Home assistant · TIAGo Pro



loading the navigation map

reMap → knowledge base
+ kitchen, living_room, corridor
rdf:type Zone

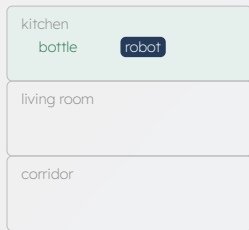
LLM context and execution

S You are [...]. Available rooms: kitchen,
living_room, corridor.



Trace of a single request: exploration and request

Home assistant · TIAGo Pro



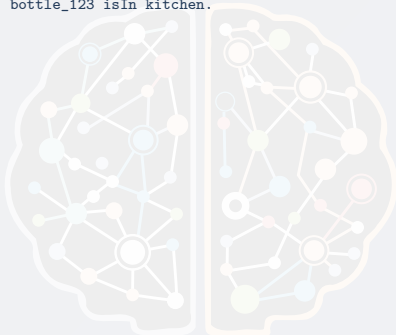
exploring the environment

reMap → knowledge base

```
+ kitchen, living_room, corridor
rdf:type Zone
+ myself isIn kitchen
+ bottle_123 rdf:type Bottle
+ myself sees bottle_123
```

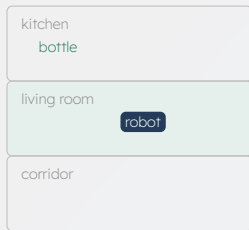
LLM context and execution

```
S You are [...]. Available rooms: kitchen,
living_room, corridor.
S myself isIn kitchen, sees bottle_123.
bottle_123 isIn kitchen.
```



Trace of a single request: exploration and request

Home assistant · TIAGo Pro



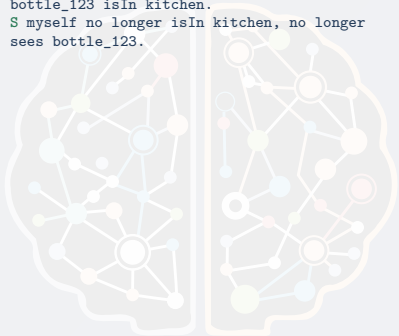
exploring the environment

reMap → knowledge base

```
+ kitchen, living_room, corridor
rdf:type Zone
+ myself isIn kitchen
+ bottle_123 rdf:type Bottle
+ myself sees bottle_123
- myself isIn kitchen
- myself sees bottle_123
```

LLM context and execution

```
S You are [...]. Available rooms: kitchen,
living_room, corridor.
S myself isIn kitchen, sees bottle_123.
bottle_123 isIn kitchen.
S myself no longer isIn kitchen, no longer
sees bottle_123.
```



Trace of a single request: exploration and request

Home assistant · TIAGo Pro



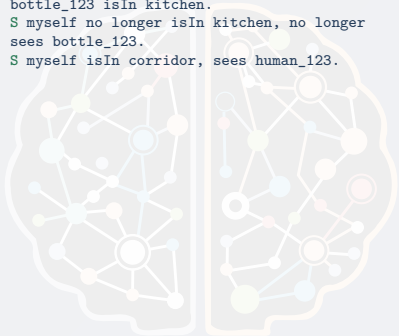
exploring the environment

reMap → knowledge base

```
+ kitchen, living_room, corridor
rdf:type Zone
+ myself isIn kitchen
+ bottle_123 rdf:type Bottle
+ myself sees bottle_123
- myself isIn kitchen
- myself sees bottle_123
+ myself isIn corridor
+ human_123 rdf:type Human
+ myself sees human_123
```

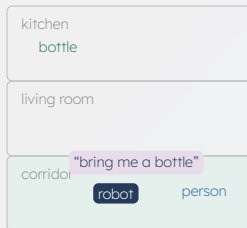
LLM context and execution

```
S You are [...]. Available rooms: kitchen,
living_room, corridor.
S myself isIn kitchen, sees bottle_123.
bottle_123 isIn kitchen.
S myself no longer isIn kitchen, no longer
sees bottle_123.
S myself isIn corridor, sees human_123.
```



Trace of a single request: exploration and request

Home assistant · TIAGo Pro



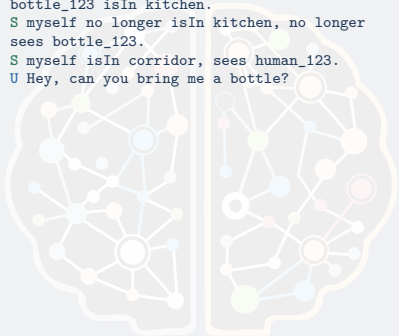
listening

reMap → knowledge base

```
+ kitchen, living_room, corridor
rdf:type Zone
+ myself isIn kitchen
+ bottle_123 rdf:type Bottle
+ myself sees bottle_123
- myself isIn kitchen
- myself sees bottle_123
+ myself isIn corridor
+ human_123 rdf:type Human
+ myself sees human_123
```

LLM context and execution

```
S You are [...]. Available rooms: kitchen,
living_room, corridor.
S myself isIn kitchen, sees bottle_123.
bottle_123 isIn kitchen.
S myself no longer isIn kitchen, no longer
sees bottle_123.
S myself isIn corridor, sees human_123.
U Hey, can you bring me a bottle?
```



Trace of a single request: exploration and request

Home assistant · TIAGo Pro



LLM extracting intent and plan

reMap → knowledge base

```
+ kitchen, living_room, corridor
rdf:type Zone
+ myself isIn kitchen
+ bottle_123 rdf:type Bottle
+ myself sees bottle_123
- myself isIn kitchen
- myself sees bottle_123
+ myself isIn corridor
+ human_123 rdf:type Human
+ myself sees human_123
```

LLM context and execution

```
S You are [...]. Available rooms: kitchen,
living_room, corridor.
S myself isIn kitchen, sees bottle_123.
bottle_123 isIn kitchen.
S myself no longer isIn kitchen, no longer
sees bottle_123.
S myself isIn corridor, sees human_123.
U Hey, can you bring me a bottle?
A {say: "Sure!", intent: {plan:
navigate(kitchen), look_for_human(),
ask_human_for_help("can you put a bottle on
my tray?"), navigate(corridor), say("here
it is")}}
```

Trace of a single request: exploration and request

Home assistant · TIAGo Pro



mission controller: plan
validation

reMap → knowledge base

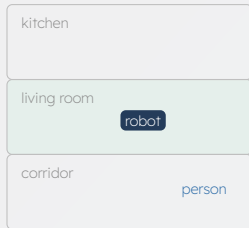
```
+ kitchen, living_room, corridor
rdf:type Zone
+ myself isIn kitchen
+ bottle_123 rdf:type Bottle
+ myself sees bottle_123
- myself isIn kitchen
- myself sees bottle_123
+ myself isIn corridor
+ human_123 rdf:type Human
+ myself sees human_123
```

LLM context and execution

```
S You are [...]. Available rooms: kitchen,
living_room, corridor.
S myself isIn kitchen, sees bottle_123.
bottle_123 isIn kitchen.
S myself no longer isIn kitchen, no longer
sees bottle_123.
S myself isIn corridor, sees human_123.
U Hey, can you bring me a bottle?
A {say: "Sure!", intent: {plan:
navigate(kitchen), look_for_human(),
ask_human_for_help("can you put a bottle on
my tray?"), navigate(corridor), say("here
it is")}}
S plan validated: kitchen is a known
navigable location, every skill exists
```

Trace of a single request: execution

Home assistant · TIAGo Pro



executing `navigate(kitchen)`

reMap → knowledge base

— myself isIn corridor

LLM context and execution



Trace of a single request: execution

Home assistant · TIAGo Pro



reMap → knowledge base

```
— myself isIn corridor  
+ myself isIn kitchen  
+ human_456 rdf:type Human  
+ myself sees human_456
```

LLM context and execution

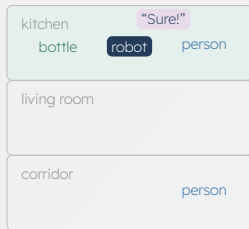
```
S myself isIn kitchen, sees human_456.  
A {say: "Can you put a bottle on my tray?"}
```

executing ask_human_for_help



Trace of a single request: execution

Home assistant · TIAGo Pro



human agreed, bottle placed
on the tray

reMap → knowledge base

- myself isIn corridor
- + myself isIn kitchen
- + human_456 rdf:type Human
- + myself sees human_456

LLM context and execution

S myself isIn kitchen, sees human_456.
A {say: "Can you put a bottle on my tray?"}
U Sure!



Trace of a single request: execution

Home assistant · TIAGo Pro



executing `navigate(corridor)`

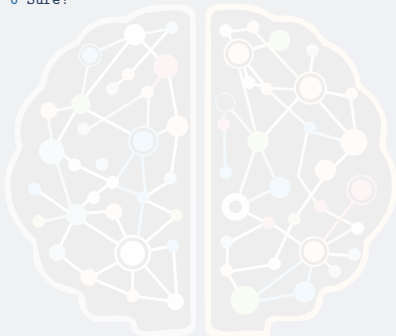
reMap → knowledge base

```
— myself isIn corridor  
+ myself isIn kitchen  
+ human_456 rdf:type Human  
+ myself sees human_456  
+ myself isIn corridor  
+ myself sees human_789
```

The person in the corridor is asserted under a new identifier: the home scenario performs no face or voice re-identification.

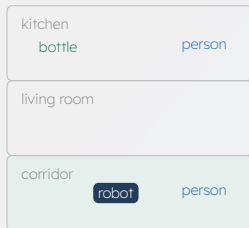
LLM context and execution

```
S myself isIn kitchen, sees human_456.  
A {say: "Can you put a bottle on my tray?"}  
U Sure!
```



Trace of a single request: execution

Home assistant · TIAGo Pro



executing say("Here you are.")

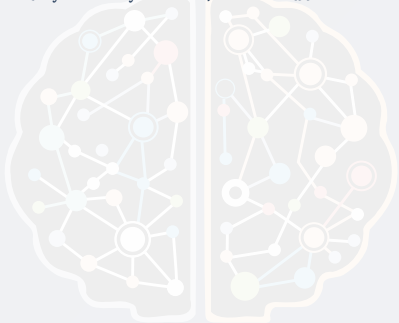
reMap → knowledge base

```
— myself isIn corridor
+ myself isIn kitchen
+ human_456 rdf:type Human
+ myself sees human_456
+ myself isIn corridor
+ myself sees human_789
```

The person in the corridor is asserted under a new identifier: the home scenario performs no face or voice re-identification.

LLM context and execution

```
S myself isIn kitchen, sees human_456.
A {say: "Can you put a bottle on my tray?"}
U Sure!
A {say: "Here you are.", intent: {}}
```



Trace of a single request: execution

Home assistant · TIAGo Pro



done

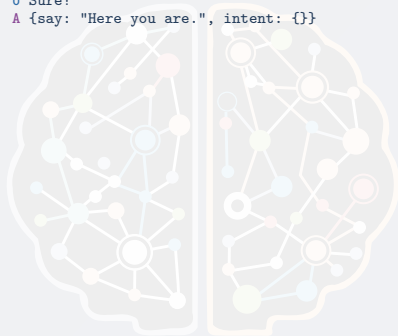
reMap → knowledge base

```
— myself isIn corridor  
+ myself isIn kitchen  
+ human_456 rdf:type Human  
+ myself sees human_456  
+ myself isIn corridor  
+ myself sees human_789
```

The person in the corridor is asserted under a new identifier: the home scenario performs no face or voice re-identification.

LLM context and execution

```
S myself isIn kitchen, sees human_456.  
A {say: "Can you put a bottle on my tray?"}  
U Sure!  
A {say: "Here you are.", intent: {}}
```



Verbal response and plan in one structured response

LLM output for the utterance "can you bring me a bottle?"

```
{
  "say": "Sure, I'll get your bottle from the kitchen.",
  "user_intent": {
    "type": "__intent_start_activity__",
    "object": "bring_bottle_from_kitchen",
    "plan": [
      {"skill": "navigate",
       "params": {"goal": "kitchen"}},
      {"skill": "look_for_human",
       "params": {"patterns": []}},
      {"skill": "ask_human_for_help",
       "params": {"text": "Can you put a bottle on my tray?",
                  "recipient": "?look_for_human.found_entities"}},
      {"skill": "navigate",
       "params": {"goal": "living_room"}},
      {"skill": "say",
       "params": {"input": "Here is your bottle."}}
    ]
  }
}
```



Verbal response and plan in one structured response

LLM output for the utterance "can you bring me a bottle?"

```
{
  "say": "Sure, I'll get your bottle from the kitchen.",
  "user_intent": {
    "type": "__intent_start_activity__",
    "object": "bring_bottle_from_kitchen",
    "plan": [
      {"skill": "navigate",
       "params": {"goal": "kitchen"}},
      {"skill": "look_for_human",
       "params": {"patterns": []}},
      {"skill": "ask_human_for_help",
       "params": {"text": "Can you put a bottle on my tray?",
                  "recipient": "?look_for_human.found_entities"}},
      {"skill": "navigate",
       "params": {"goal": "living_room"}},
      {"skill": "say",
       "params": {"input": "Here is your bottle."}}
    ]
  }
}
```

- **Room inference.** The target room is not stated in the utterance. It is inferred from the environment system prompts accumulated while the robot explored the environment.

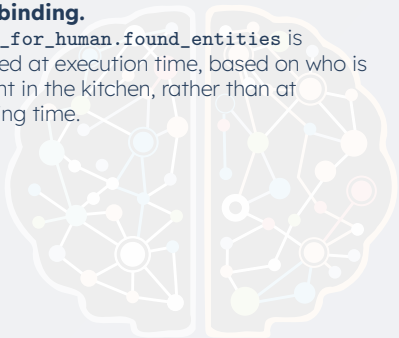


Verbal response and plan in one structured response

LLM output for the utterance "can you bring me a bottle?"

```
{
  "say": "Sure, I'll get your bottle from the kitchen.",
  "user_intent": {
    "type": "__intent_start_activity__",
    "object": "bring_bottle_from_kitchen",
    "plan": [
      {"skill": "navigate",
       "params": {"goal": "kitchen"}},
      {"skill": "look_for_human",
       "params": {"patterns": []}},
      {"skill": "ask_human_for_help",
       "params": {"text": "Can you put a bottle on my tray?",
                  "recipient": "?look_for_human.found_entities"}},
      {"skill": "navigate",
       "params": {"goal": "living_room"}},
      {"skill": "say",
       "params": {"input": "Here is your bottle."}}
    ]
  }
}
```

- **Room inference.** The target room is not stated in the utterance. It is inferred from the environment system prompts accumulated while the robot explored the environment.
- **Late binding.** ?look_for_human.found_entities is resolved at execution time, based on who is present in the kitchen, rather than at planning time.

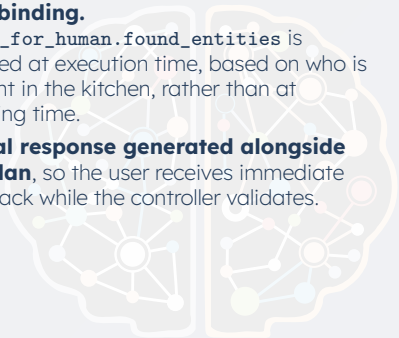


Verbal response and plan in one structured response

LLM output for the utterance "can you bring me a bottle?"

```
{
  "say": "Sure, I'll get your bottle from the kitchen.",
  "user_intent": {
    "type": "__intent_start_activity__",
    "object": "bring_bottle_from_kitchen",
    "plan": [
      {"skill": "navigate",
       "params": {"goal": "kitchen"}},
      {"skill": "look_for_human",
       "params": {"patterns": []}},
      {"skill": "ask_human_for_help",
       "params": {"text": "Can you put a bottle on my tray?",
                  "recipient": "?look_for_human.found_entities"}},
      {"skill": "navigate",
       "params": {"goal": "living_room"}},
      {"skill": "say",
       "params": {"input": "Here is your bottle."}}
    ]
  }
}
```

- **Room inference.** The target room is not stated in the utterance. It is inferred from the environment system prompts accumulated while the robot explored the environment.
- **Late binding.** ?look_for_human.found_entities is resolved at execution time, based on who is present in the kitchen, rather than at planning time.
- **Verbal response generated alongside the plan,** so the user receives immediate feedback while the controller validates.



Verbal response and plan in one structured response

LLM output for the utterance "can you bring me a bottle?"

```
{
  "say": "Sure, I'll get your bottle from the kitchen.",
  "user_intent": {
    "type": "__intent_start_activity__",
    "object": "bring_bottle_from_kitchen",
    "plan": [
      {"skill": "navigate",
       "params": {"goal": "kitchen"}},
      {"skill": "look_for_human",
       "params": {"patterns": []}},
      {"skill": "ask_human_for_help",
       "params": {"text": "Can you put a bottle on my tray?",
                  "recipient": "?look_for_human.found_entities"}},
      {"skill": "navigate",
       "params": {"goal": "living_room"}},
      {"skill": "say",
       "params": {"input": "Here is your bottle."}}
    ]
  }
}
```

- **Room inference.** The target room is not stated in the utterance. It is inferred from the environment system prompts accumulated while the robot explored the environment.
- **Late binding.** `?look_for_human.found_entities` is resolved at execution time, based on who is present in the kitchen, rather than at planning time.
- **Verbal response generated alongside the plan,** so the user receives immediate feedback while the controller validates.

The plan checker then verifies that **kitchen** is a known navigable location and that every skill in the sequence is available, before execution starts.

Limitations and future work

Open directions

Distilled on-device models

We plan to evaluate distilled models running on the robot, to remove the network dependency from intent extraction and dialogue and to characterise the resulting trade-off in reasoning quality and latency.



Limitations and future work

Open directions

Distilled on-device models

We plan to evaluate distilled models running on the robot, to remove the network dependency from intent extraction and dialogue and to characterise the resulting trade-off in reasoning quality and latency.

Learned policies as skills

Physical interaction is currently delegated to scripted navigation and tray-transport skills. Integrating learned VLA policies as skills under symbolic supervision would extend manipulation while preserving the inspectable decision layer.



Limitations and future work

Open directions

Distilled on-device models

We plan to evaluate distilled models running on the robot, to remove the network dependency from intent extraction and dialogue and to characterise the resulting trade-off in reasoning quality and latency.

Controlled longitudinal study

We report operational deployments rather than a study. Task success, user acceptance and the perceived usefulness of the generated explanations remain to be measured.

Learned policies as skills

Physical interaction is currently delegated to scripted navigation and tray-transport skills. Integrating learned VLA policies as skills under symbolic supervision would extend manipulation while preserving the inspectable decision layer.



Limitations and future work

Open directions

Distilled on-device models

We plan to evaluate distilled models running on the robot, to remove the network dependency from intent extraction and dialogue and to characterise the resulting trade-off in reasoning quality and latency.

Controlled longitudinal study

We report operational deployments rather than a study. Task success, user acceptance and the perceived usefulness of the generated explanations remain to be measured.

Learned policies as skills

Physical interaction is currently delegated to scripted navigation and tray-transport skills. Integrating learned VLA policies as skills under symbolic supervision would extend manipulation while preserving the inspectable decision layer.

Per-agent knowledge models

KnowledgeCore supports separate knowledge models per agent. Perspective taking in multi-party interaction is supported by the design but not exercised in the current deployments.