Development of an Action Planning System Using Explainable AI for Home Service Robots

Takashi Akamatsu
Department of Creative Engineering, NIT, Kitakyushu College,
5-20-1 Shii, Kokuraminamiku, Kitakyushu, Fukuoka, 802-0985, Japan

Yuma Yoshimoto*
Department of Creative Engineering, NIT, Kitakyushu College,
5-20-1 Shii, Kokuraminamiku, Kitakyushu, Fukuoka, 802-0985, Japan

Email: k21002ta@apps.kct.ac.jp, yoshimoto@kct.ac.jp
*Corresponding Author

Abstract

In recent years, the demand for home service robots has increased due to the progression of the declining birthrate and aging population. For home deployment, the Command Understanding Task (CUT)—in which a robot generates an action plan from a spoken command—is essential. While large language models (LLM) offer flexible plan generation, their opaque reasoning makes error analysis and correction difficult. In this research, we propose two methods to achieve both explainability and flexibility: (1) an ontology-based action planning approach and (2) an LLM-based automatic ontology construction method. The former enables transparent plans by reasoning over phrases extracted from commands and knowledge defined in the ontology. The latter shows that LLM can automatically generate RDF-based ontologies, including object categories, shapes, and spatial relations. The results demonstrate the feasibility of combining explainable planning with automatic knowledge expansion. Future work will extend inference capabilities and support large-scale ontology construction to handle more complex commands.

Keywords: Home Service Robot, Ontology, Explainable AI (XAI)

1. Introduction

With the progression of population aging and declining birth rates, the demand for home service robots has been increasing. When deploying robots into the home, it is essential to achieve "Command Understanding Task (CUT)" where the robot autonomously generates an action plan from verbal commands given by the user and executes the commands according to that plan.

In recent years, large language models (LLM) have been increasingly applied to a wide range of processes in robotics due to their high flexibility. Consequently, the use of LLM for action plan generation in CUT, where flexibility is crucial, is becoming a mainstream approach [1]. However, because the internal reasoning processes of LLM are opaque and lack explainability, it is difficult to identify or correct the causes of errors when a robot behaves according to an incorrect plan, potentially leading to accidents or malfunctions. This issue is considered one of the major obstacles to the deployment of home service robots into home environments.

In this research, we aim to develop an action plan generation system that achieves both flexibility and explainability by leveraging explainable AI (XAI) ontologies.

2. Related Research

2.1. *Ontology [2]*

2.1.1. *Overview*

An ontology is a formal specification of concepts, relations, and constraints within a target domain, described as machine-readable knowledge. When combined with a reasoning engine, it becomes a type of explainable AI (XAI) capable of performing logical inference.

2.1.2. *Constituent Elements*

An ontology is composed of the following elements.

1. Classes
   A set of instances that share common characteristics within a domain.
   Example: Apple
2. Individual (Instances)
   A concrete entity that belongs to a class.
   Example: an apple on a desk
3. Object Properties (Relations)
   A semantic connection between classes or between instances.
   Example: apple $\subset$ fruit
4. Axioms

© The 2026 International Conference on Artificial Life and Robotics (ICAROB2026)

A logical constraint or rule over classes and relations.
Example: fruit that is red and round = apple

2.2. *Protégé [3]*

An open-source ontology editing tool developed by Stanford University. It enables users to visually construct ontologies that would otherwise need to be described manually in RDF format. In this research, it is used both for manually constructing ontologies and for visualizing ontologies generated by LLM.

2.3. *Human Support Robot (HSR) [4]*

HSR is a mobile manipulator designed for home environments, equipped with functions for manipulation tasks such as object grasping, as well as communication capabilities using a microphone and speakers. It is being developed with the aim of performing tasks including furniture manipulation, transporting daily items, and tidying up rooms. Figure 1 shows the appearance of the HSR. In this research, we use HSR as our research platform.

Figure 1 Appearance of HSR

3. Proposed Method

To achieve action plan generation that balances flexibility and explainability, we propose two systems: Ontology-Based Action Plan Generation and Automatic Ontology Construction Using LLM.

3.1. *Ontology-Based Action Plan Generation*

The overall processing flow of Ontology-Based Action Plan Generation is shown in Figure 2. First, the system obtains the user command using speech recognition (Step ①). Next, it extracts relevant words from the command using an LLM (GPT-4o) (Step ②). Finally, based on the extracted words and the knowledge defined in the ontology, the system generate the action plan (Step ③). The details of Step ③ are shown in Figure 3. Specifically, (a) the system first refers to the relations between instances described in the ontology and retrieves the necessary actions corresponding to the verb extracted from the command, generating the basis of the action plan. Then, (b) reasoning is performed as needed, and each action is finalized based on the reasoning results.

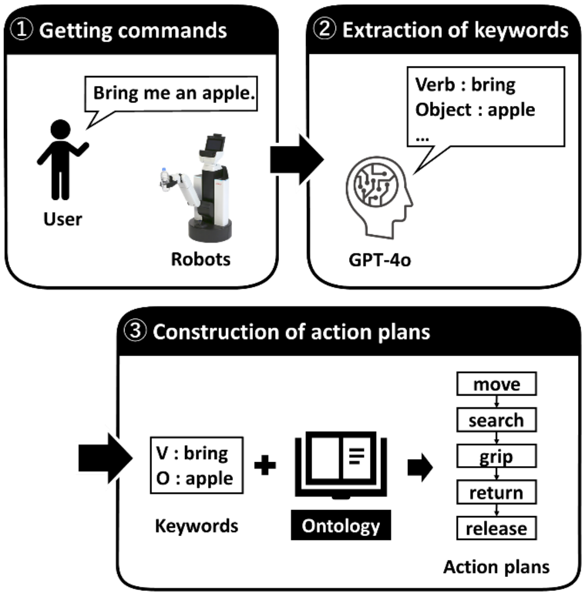


Figure 2 Processing Flow of Ontology-Based Action Plan Generation

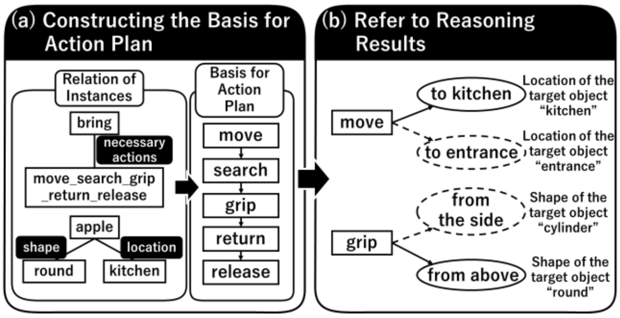


Figure 3 Flow of Action Plan Generation

3.2. *Automatic Ontology Construction Using LLM*

3.2.1. *Problem of Ontology-Based Action-Plan Generation*

The primary advantage of Ontology-Based Action Plan Generation lies in its high explainability, which facilitates the identification and correction of the causes of accidents or malfunctions. However, accommodating a wide variety of commands requires defining an extensive amount of knowledge, and manually constructing such ontologies has limitations. Therefore, to address this issue of flexibility, we employ LLM to automatically construct ontologies.

3.2.2. *Process of Automated Ontology Construction*

The process by which the LLM automatically constructs an ontology follows the six-step pipeline illustrated in Figure 4. In Step ① (Class Hierarchy Design), a class hierarchy is defined that distinguishes entity, attribute, and context classes. In Step ② (Relation Design), binary relations between these classes are specified together with their domains, ranges, and cardinality constraints. In Step ③ (Instance Definitions), prototype instances are introduced for each attribute and context class. Step ④a (Object Classes and Instance Generation) defines concrete entity classes and generates their instances according to a unified naming rule. In Step ④b (Assignment of Relation), each entity instance is associated with exactly one attribute and one context by means of the predefined relations that refer to the corresponding prototypes. Finally, Step ⑤ (OWL (RDF/XML) Export) serializes the resulting ontology—including all classes, relations, instances, and assertions—into OWL (RDF/XML) format so that it can be directly used for ontology-based action-plan generation.

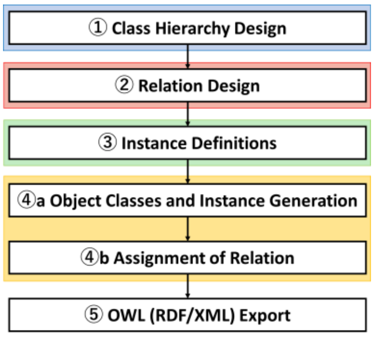


Figure 4 Flowchart of LLM-Based Ontology Construction

4. Experiment

4.1. *Ontology-Based Action Plan Generation Experiment*

4.1.1. *Experimental Procedure*

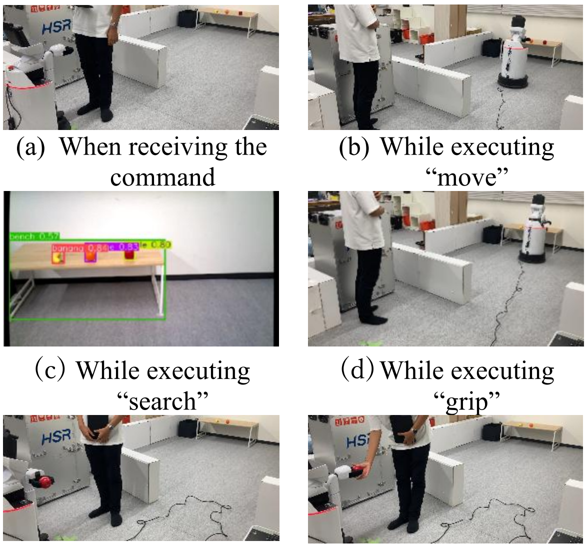
This experiment was conducted according to the following procedure.

1. The command “Bring me an apple.” was given to the robot, and an action plan was generated based on the ontology.
2. The robot was then operated in accordance with the generated action plan.

4.1.2. *Experimental Results*

The reasoning result is shown in Figure 5, and the sequence of robot actions is shown in Figure 6. First, as shown in Figure 6(a), the robot received the command, generated an action plan based on the actions required for the verb “bring,” and performed reasoning. Next, as shown in Figure 6(b), the action “move” was executed, and the robot moved to the kitchen (Move_to_kitchen) according to the reasoning result in Figure 5. Then, as shown in Figure 6(c), the action “search” was executed, and the three-dimensional position of the “apple” was obtained via object recognition. Furthermore, as shown in Figure 6(d), the action “grip” was executed, and the apple was grasped from above (Grip_from_above) based on the reasoning result in Figure 5. After that, as shown in Figure 6(e), the action “return” was executed and the robot returned to its initial position. Finally, as shown in Figure 6(f), the action “release” was executed and the “apple” was handed to the user.

Figure 5 Reasoning Results



(a) When receiving the command
(b) While executing “move”
(c) While executing “search”
(d) While executing “grip”
(e) While executing “return”
(f) While executing “release”

Figure 6 Sequence of Robot Actions

4.2. *LLM-Based Automatic Ontology Construction Experiment*

4.2.1. *Experimental Details*

We used an LLM (GPT-5.1) to automatically construct the ontology for some objects included in the YCB object set.

4.2.2. *Construction Image*

Figure 7 illustrates the intended structure of the classes and instances to be automatically constructed by the LLM in this research. First, the parent class Object is defined, and subclasses such as Food_item and Kitchen_item are introduced beneath it. Furthermore, subclasses corresponding to each object name, such as Apple and Knife, are defined, and instances such as apple and knife that belong to these subclasses are generated.

Similarly, for Shape and Location, we define shape classes such as Round and Cylinder, and location classes such as Entrance, Kitchen, and Living_room, and generate the corresponding instances round, cylinder, entrance, kitchen, and living room.

Table 1 lists the relation names between instances that are to be defined by the LLM, together with their meanings. As an example, we assume that the instance apple is assigned round as the value of the relation has_shape and kitchen as the value of the relation in_the.

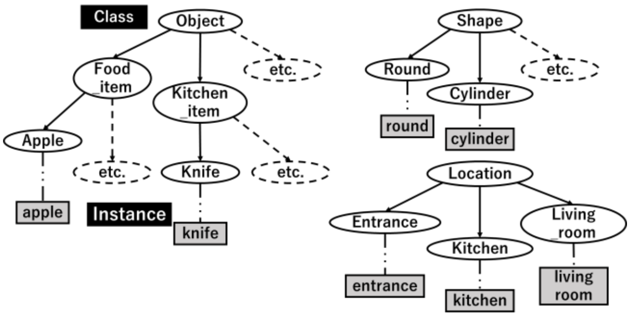


Figure 7 Construction Image

Table 1 Relation Between Instances

Relation Names	Meanings
has_shape	the shape of an object
in_the	the location of an object

4.2.3. *Prompt for Automatic Ontology Construction*

Figure 8 shows the prompt used in this experiment. The explanations corresponding to the numbered labels in the figure are given below.

4.2.3.1. *① Class structure*

We define three top-level classes: Object, Shape, and Location. Under Object, we introduce category classes such as Food_item, Toy_item, Kitchen_item, Tool_item, and Sports_item. Under Shape, we define shape classes such as Round and Cube, and under Location we define place classes such as Kitchen, Living_room, and Entrance.

4.2.3.2. *② Relation definitions*

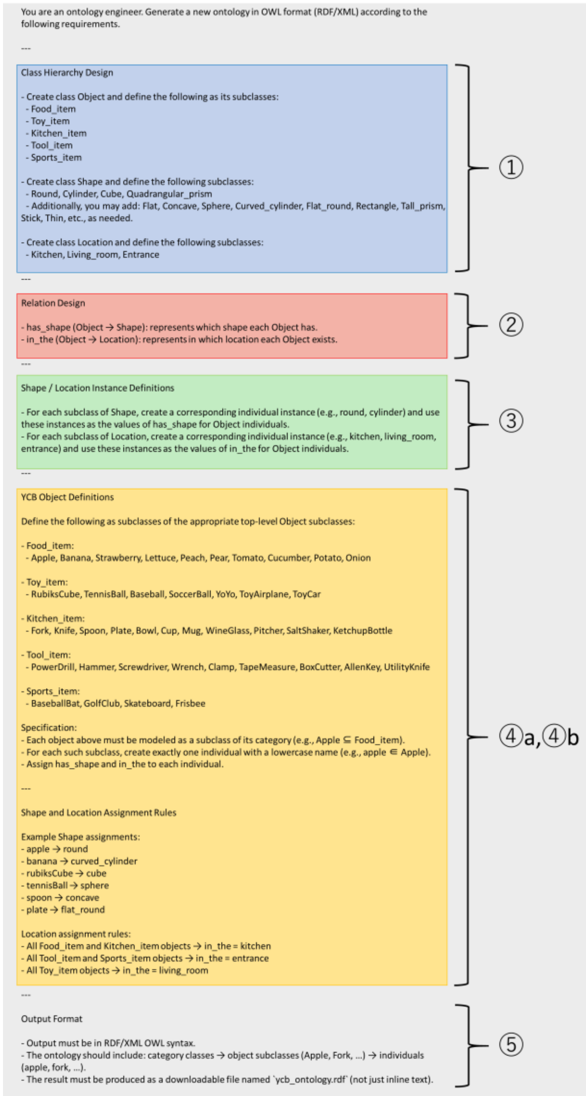
We define two relations that connect objects to shapes and locations: has_shape (from Object to Shape) and in_the (from Object to Location).

4.2.3.3. *③ Shape and Location instance definitions*

For each Shape and Location class, we create a corresponding lowercase instance (e.g., round, cube, kitchen) and use these instances as the values of the has_shape and in_the relations for object instances.

4.2.3.4. *④ Definition of YCB objects*

Based on the YCB objects, we define classes such as Apple and Fork as subclasses of the appropriate category classes (e.g., Apple ⊆ Food_item, Fork ⊆ Kitchen_item) and create exactly one lowercase instance for each class (apple, fork, etc.). Each object instance is then assigned both a has_shape relation (e.g., apple → round, banana → curved_cylinder, rubikscube → cube) and an in_the relation according to category-based rules: all Food_item and Kitchen_item objects are placed in kitchen, all Tool_item and Sports_item objects in entrance, and all Toy_item objects in living_room.



You are an ontology engineer. Generate a new ontology in OWL format (RDF/XML) according to the following requirements.

\---

Class Hierarchy Design

- Create class Object and define the following as its subclasses:
  - Food_item
  - Toy_item
  - Kitchen_item
  - Tool_item
  - Sports_item

- Create class Shape and define the following subclasses:
  - Round, Cylinder, Cube, Quadrangular_prism
  - Additionally, you may add: Flat, Concave, Sphere, Curved_cylinder, Flat_round, Rectangle, Tall_prism, Stick, Thin, etc., as needed.

- Create class Location and define the following subclasses:
  - Kitchen, Living_room, Entrance

①

\---

Relation Design

- has_shape (Object → Shape): represents which shape each Object has.
- in_the (Object → Location): represents in which location each Object exists.

②

\---

Shape / Location Instance Definitions

- For each subclass of Shape, create a corresponding individual instance (e.g., round, cylinder) and use these instances as the values of has_shape for Object individuals.
- For each subclass of Location, create a corresponding individual instance (e.g., kitchen, living_room, entrance) and use these instances as the values of in_the for Object individuals.

③

\---

YCB Object Definitions

Define the following as subclasses of the appropriate top-level Object subclasses:

- Food_item:
  - Apple, Banana, Strawberry, Lettuce, Peach, Pear, Tomato, Cucumber, Potato, Onion

- Toy_item:
  - RubiksCube, TennisBall, Baseball, SoccerBall, YoYo, ToyAirplane, ToyCar

- Kitchen_item:
  - Fork, Knife, Spoon, Plate, Bowl, Cup, Mug, WineGlass, Pitcher, SaltShaker, KetchupBottle

- Tool_item:
  - PowerDrill, Hammer, Screwdriver, Wrench, Clamp, TapeMeasure, BoxCutter, AllenKey, UtilityKnife

- Sports_item:
  - BaseballBat, GolfClub, Skateboard, Frisbee

Specification:
- Each object above must be modeled as a subclass of its category (e.g., Apple ⊆ Food_item).
- For each such subclass, create exactly one individual with a lowercase name (e.g., apple ∈ Apple).
- Assign has_shape and in_the to each individual.

\---

Shape and Location Assignment Rules

Example Shape assignments:
- apple → round
- banana → curved_cylinder
- rubiksCube → cube
- tennisBall → sphere
- spoon → concave
- plate → flat_round

Location assignment rules:
- All Food_item and Kitchen_item objects → in_the = kitchen
- All Tool_item and Sports_item objects → in_the = entrance
- All Toy_item objects → in_the = living_room

④a,④b

\---

Output Format

- Output must be in RDF/XML OWL syntax.
- The ontology should include: category classes → object subclasses (Apple, Fork, ...) → individuals (apple, fork, ...).
- The result must be produced as a downloadable file named `ycb_ontology.rdf` (not just inline text).

⑤

Figure 8 Prompt Used in the Experiment

4.2.4. *Experimental Results*

A subset of the automatic construction results is shown in Figures 9 and 10. From Figure 9, it was confirmed that the classes were constructed consistently with the conceptual structure illustrated in Figure 7. Furthermore, Figure 10 shows that the relations between instances were assigned as intended.

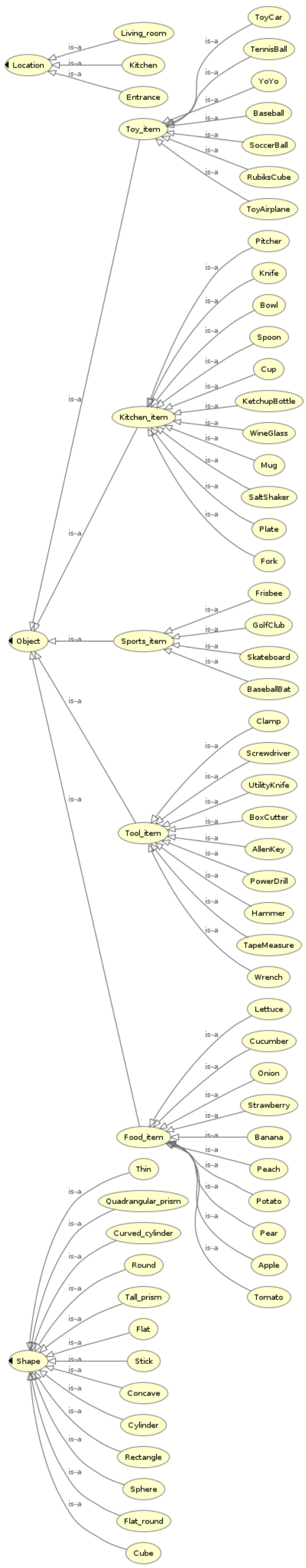


Figure 9 Class Construction Results

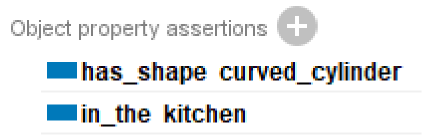


Figure 10 Relations of the Instance "banana"

5. Conclusion

In this research, we confirmed that the ontology-based action plan generation was effective for basic household commands. Furthermore, we examined the feasibility of automatic ontology construction using LLM and demonstrated that fundamental components such as class, relation, and instance definitions can be generated automatically. In the future, we aim to extend this approach to the automatic construction of large-scale ontologies that include the axioms required for reasoning, so that the system can handle more complex and diverse commands assumed in real home environments. Ultimately, our goal is to enable the execution of advanced commands generated by the GPSR command generator used in RoboCup@Home [5], and to realize an action planning system that is both explainable and flexible.

References

1. Kosei, Y. et al., "Development of a SayCan-based task planning system capable of handling abstract nouns," ICAROB2024, vol. 29, pp. 430-434, 2024.
2. T. Slimani, "A Study Investigating Typical Concepts and Guidelines for Ontology Building," Journal of Emerging Trends in Computing and Information Sciences, vol. 5, No. 12, pp. 886–893, 2014.
3. protégé, "protégé," https://protege.stanford.edu/, 2025.12.08 accessed.
4. T. Yamamoto, et al.: "Development of Human Support Robot as the research platform of a domestic mobile manipulator," ROBOMECH Journal, vol. 6, article no. 4, pp. 1–15, 2019.
5. RoboCup@Home Technical Committee, "RoboCup@Home Rules & Regulations," https://robocupathome.github.io/RuleBook/rulebook/master.pdf, 2025.12.08 accessed.

Authors Introduction

Mr. Takashi Akamatsu

He enrolled at National Institute of Technology (KOSEN), Kitakyushu College, Japan, in 2021. In 2023, he pursued the Information Systems course, focusing on algorithms and control. He commenced his research in robotics in 2024.

Dr. Yuma Yoshimoto

He received his B.Eng. degree from National Institute of Technology (KOSEN), Maizuru College, Japan, in 2016. He received his M.Eng. and D. Eng. degrees from Kyushu Institute of Technology, Japan, in 2018 and 2021, respectively. And he was JSPS researcher, in 2019 - 2021. He was a postdoctoral researcher at the Kyushu Institute of Technology, Japan in 2021-2022. Currently, he is assistant professor at the National Institute of Technology (KOSEN), Kitakyushu College, Japan. His research interests include deep learning, robot vision and digital hardware design. He is a member of IEICE, IEEE.