A Knowledge-based Multi-entity and Cooperative System Architecture

Manuel Mühlig, Lydia Fischer, Stephan Hasler and Jörg Deigmöller[1]

***Abstract*—Future intelligent systems will become more complex and they will be composed of potentially very different artificial agents like mobile robots, smart home infrastructure, and smartphones that collect data and that have capabilities to perform certain tasks. The challenges will be that the elements such a system is composed of will not be known in advance and might change dynamically. Further, the tasks that need to be fulfilled will be unknown beforehand and might require cooperation with humans to make use of their abilities. In this paper we provide a description of a system with which we try to tackle these challenges. Our system uses available resources (robots, humans) and the actual state of the environment to provide a plan in order to satisfy a given request. We demonstrate the flexibility of our system through varying the available resources and the state of the environment. To our knowledge there are few approaches only that (i) explicitly model and use humans together with intelligent agents in the same plan in this kind of abstraction level, and (ii) that have a system using heterogeneous agents at the same time.**

I. Introduction

When looking at the current trend of intelligent systems, we assume they will become more complex and will be composed of multiple artificial agents. Those agents are potentially very different in their form and functionality, such as mobile robots, static smart home infrastructure, or smartphones. This raises new challenges that need to be addressed. For example, since the system elements are likely not known in advance and might change from user to user or over time, future intelligent systems need to be dynamic in this respect. Furthermore, these systems should be embedded in the real world and interact with humans, which means that human-understandable concepts about the environment need to be employed [1], [2]. A viable approach for this is to use semantic knowledge, however with its known problems towards generalizability and grounding [3]. Besides that, the designer of such systems will most-likely not consider all future tasks that ought to be fulfilled. The systems therefore need to be extendable over time and, even better, cooperate with humans to compensate for lacking abilities, as for example already successfully shown in well-defined assembly tasks [4], [5]. This is a promising way for elevating an assistant system from a simple tool towards a cooperation partner [6].

In this paper we provide a description of a system with which we tackle some of the mentioned challenges. We are targeting at a system that operates in an office environment and which uses multiple heterogeneous resources like mobile robots, humans, and smart rooms in order to perform tasks, such as fetching objects or guiding visitors. As a testbed, we have implemented the so-called *Living Lab* as part of our normal office workspace. A part of the office space has been equipped with cameras and other sensors, mobile robots can navigate around, and a smart room, the *Smart Lobby*, has been installed. In this setting, we demonstrate the flexibility of our system by varying the available resources (robots and persons) and the state of the environment. To our knowledge there are only few approaches that (i) explicitly model and use humans together with intelligent agents in the same plan in this kind of abstraction level, and (ii) that have a system using heterogeneous agents at the same time.

The key features of the presented system are:

1) a centralized system architecture that allows for dynamic registration of heterogeneous entities.
2) a shared knowledge representation including semantic and model information.
3) the automatic generation of multi-entity planning problems given the capabilities of the registered entities.
4) the explicit modelling of persons as part of the system and their involvement in achieving goals.
5) methods for providing explanations of the system's behavior by the system itself.

The paper is organized as follows. The next section embeds our work in the literature. Then, Section III provides a brief review of the overall architecture. The main parts of the system, entity management, knowledge organization, and multi-entity planning are explained in detail in sections IV, V, and VI, respectively. Afterwards in Section VII we give an insight into our experiments and sum up in Section VIII.

II. Related Work

There are multiple related publications in the domain of service robot architectures that are relevant for this work. For example, the CMU Snackbot project [7] implemented an autonomous, interactive, service robot for snack delivery in an office space targeted at long-term operation. Also, the authors of [8] showed a service robot system that focussed on longer operation and on asking humans for help. An aspect very relevant to this paper as can be seen in later sections. A larger effort has also been conducted within the STRANDS project [9]. The single robot architecture has been proven to be stable over a long time, pursuing service tasks in interaction with naive people. Parts of their methods have been reused for the mobile robots presented in this paper, although in a new scope within our multi-entity architecture.

[1]All authors are with the Honda Research Institute Europe GmbH, Carl-Legien-Strasse 30, 63073, Offenbach/Main, Germany, {firstname.lastname}@honda-ri.de

978-1-7281-5871-6/20/$31.00

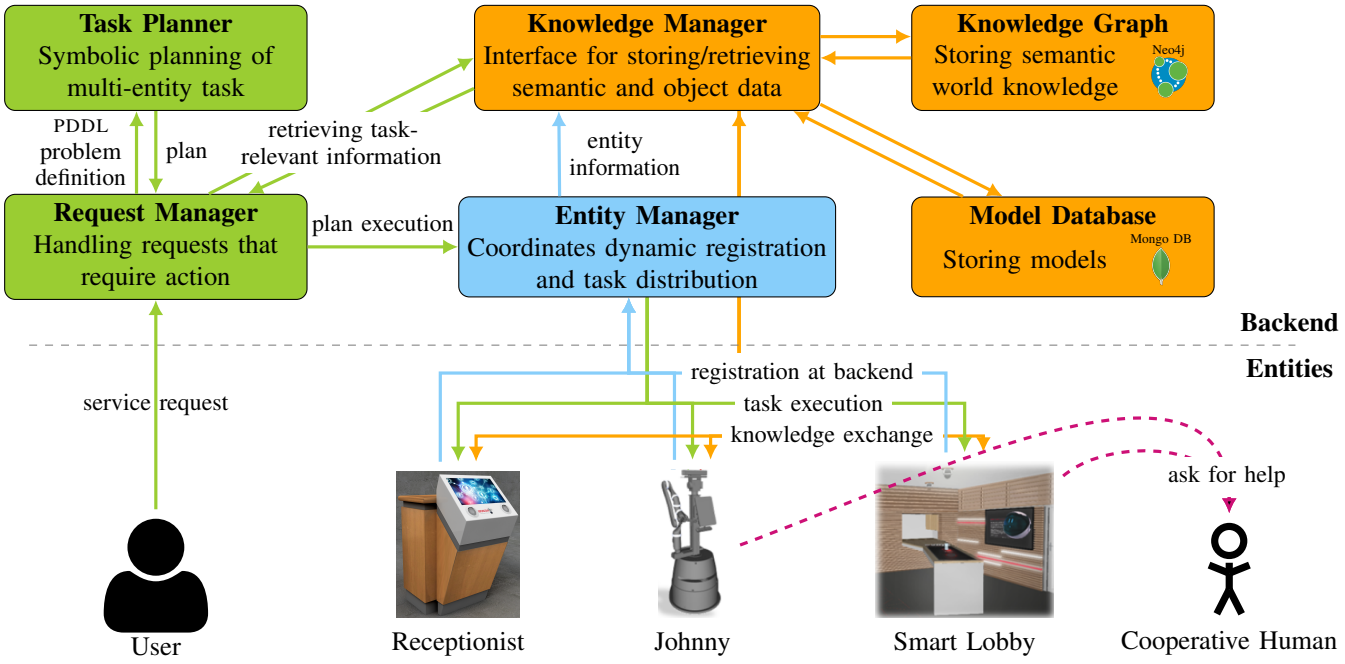

Fig. 1: The system overview shows the backend parts and the entities. The green, orange, and blue parts relate to planning, to storing and accessing the knowledge of the system, and to the registration of entities and the task distribution of created plans. Entities can be used to ask humans for help.

Since we are targeting at operating in a real office environment, the present humans need to be considered by the system. A recent overview over the field of human robot teaming and the associated challenges can be found in [10]. Also, there is earlier work considering humans during robots' actions [11], where the state of the human (e. g., standing or sitting) is considered for appropriate motion planning. We are however trying to more tightly incorporate the humans in the system's behavior. As will be explained later in this work, humans are not seen as uncontrollable constraints, but instead their capabilities are taken into account and the system might opt to *ask* the human for help.

One part of a multi-entity system is the knowledge organization and distribution amongst the components. This requires the right abstraction level due to the different sensors and capabilities of the system entities. Semantic representations are an efficient method to achieve this and a way to make knowledge gathered by single robots available to other robots [12]. As presented in [13], this seems feasible in a larger scope by using a representation that includes knowledge about the environment as well as past actions of a robot. Semantic representations also provide different ways of allowing for extendability of the knowledge under an open-world assumption. On the one hand, it allows for reasoning over unknowns, for example by incorporating the concept of hypotheses [14]. On the other hand, the semantic representation can be connected to external world knowledge. This has for example been shown in the KnowRob project [15], where information from sensory data is associated to predefined ontological information. Furthermore, in our earlier work, we also showed that such a representation is well suited for interacting with humans and for generating human-understandable explanations of a reasoning process [16]. In that previous work, the focus was more on how to represent knowledge (in particular relations between tools, actions, and objects) rather than on symbolic planning.

For planning we are building upon traditional AI methods similar to the work presented in [17]. There, the planning domain and problem (e. g., a search task in an unknown environment) is defined using a variant of the Planning Domain Definition Language (PDDL) and a plan is searched for a single robot using a combination of deterministic and decision-theoretic planners. The process is similar in our work, besides that in our case the planning problem is generated automatically according to which entities (and persons) are actually available and which capabilities they have.

III. System Architecture Overview

Figure 1 provides an overview of the components of the presented system. It is implemented as a centralized architecture separated into a backend and the entities that it controls. The term entities in this context is not restricted to robots, but also includes smart infrastructure, such as the depicted *Smart Lobby*. The later Section VI also describes how humans can be seen as part of this system in form of entities that cannot be controlled directly. Entities themselves do not communicate among each other, but only with two backend components. Firstly, the Entity Manager, which allows for entities to register at the system and for the backend to assume control of them. Secondly, the Knowledge Manager, which coordinates the storage of sensory information received from the entities and allows other components to query this information via a common interface. For actually performing a function that utilizes the registered entities, the Request Manager receives the desired goal from a user. It then queries information relevant for the planning problem using the Knowledge Manager, such

as the registered entities, their location and capabilities, and the task-relevant subset of the measured world state. This information is used to generate a sequential plan that is then executed via the Entity Manager. The system and the entities are implemented using ROS with multiple ROS masters to increase robustness wrt. wireless communication outage (based partially on the multimaster_fkie package). The subsequent three sections provide more details on the entity coordination, the knowledge representation, and the multi-entity planning, respectively.

IV. Entity Manager and Heterogeneous Entities

The system can be composed of multiple heterogeneous entities that can register and unregister themselves at the Entity Manager during runtime. For the initial registration, a UUID (Universally Unique Identifier) is being generated to identify the entity for and to be used for later re-registrations. Since the entity is formerly unknown to the system, it fully describes itself during the registration process. This information is stored in the Knowledge Graph and used later for generating plans that involve the respective entity. Entities are defined by following properties:

1) A *State Model* with the two modes *Autonomous* and *Controlled*. In the autonomous mode, the entity is known to the backend, but cannot execute a requested command. It is allowed to follow its own autonomous behavior, for example a mobile robot patrolling and interacting with people. In the controlled mode, the entity is part of a larger plan and thus not allowed to perform autonomous actions that could interfere with it. A safe switching between the autonomous and controlled mode has to be implemented for each entity specifically.
2) Entities have *Capabilities*, which are basically parametrized actions that the backend can execute with the entity. For example, changing the location of the entity or picking up an object are such capabilities. The Entity Manager can request the execution of an entity's capability only while it is in controlled mode.
3) *Sensors* are the different types of information the entity continuously provides to the backend. The term can refer to actual sensor data or to virtual (e. g., post-processed) data. For example the battery state and the pose of our mobile robots are sensors, but also is the information about spontaneously recognized persons.
4) Finally, all entities have a *Location*, which is described by a reference frame and a static (e. g., fixed camera) or dynamic (e. g., mobile robot) transformation.

After registration, an alive signal is being sent regularly to the Entity Manager in order to detect communication outages.

As can be seen from the description above, it is generally assumed that entities have an autonomous operation mode. This means, robots can autonomously move through the office space in our *Living Lab* environment. During this phase an interaction can be triggered via a wakeup word and this interaction can lead to a request to the backend system. If however the robot is in controlled mode it does not move autonomously and does not react to its wakeup word.

Fig. 2: Mobile robot Johnny during object hand over in the Smart Lobby

The following is an overview of the entities in our *Living Lab* environment. Note that only the capabilities relevant for the experiments in Section VII are mentioned here.

Mobile robots: We are using three mobile robots based on MetraLabs' SCITOS G5. It is additionally equipped with a Kinect camera mounted on a Schunk PW70 pan-tilt-unit for moving the head. The camera is encased in a custom head design. Laser scanners in the front and rear allow the robot to localize itself in the room and a Kinova JACO 2 arm enables the robot to transport objects. These robots have the capabilities *'move to location'*, and *'inform person'*. They are equipped with sensors measuring their own pose and measuring faces (recognizing persons) in a certain range. Figure 2 shows the mobile robot during the hand over of an object in the experiment described in Section VII. For more details on the robots' capabilities we refer to [18] where they have been used for interactive object learning.

Smart Lobby: The Smart Lobby [19], [20] is a room equipped with various sensors, such as Kinect cameras, microphone arrays, and touch interfaces (see Fig. 2). It contains three main screens for visualizing information or interacting with persons. One big screen on a wall is used for visualization purposes as well as for displaying a virtual avatar that represents this entity. Additionally there are two screens in the table where one of it can be used for touch interaction. The Smart Lobby can detect faces and therefore persons within the room and has the capability *'inform person'*.

Receptionist: The *Living Lab* includes a stationary computer showing the same avatar as the Smart Lobby. The computer screen is equipped with a camera and microphones and is used for introducing new persons to the system. After the introduction process, the information about the person, including name and face recognition model, is then stored in the system's knowledge representation. The Receptionist is not used during the experiment in Section VII and only mentioned here for completeness.

Persons: Persons are agents that do not register at the system. However they share semantic properties with entities

and are used by the planner (described in Section VI) as such. Persons are assumed to have all capabilities the system knows of. This assures an open-ended behavior of the system in case of requests that cannot be matched by any registered entity. One difference to normal entities however is that persons cannot be controlled directly by the Entity Manager. They need to be asked for support (i. e., executing a specific capability) by another entity.

V. Knowledge Representation

The main purpose of the knowledge representation is to provide a central location for acquired information and make the information globally accessible to all registered entities. The knowledge representation has two elements, a Knowledge Graph and a Model Database (see orange parts in Figure 1). The Knowledge Manager organizes the access to the databases. All three components are now described in more detail.

Knowledge Manager: The Knowledge Manager directs requests to the suited underlying database. To separate huge amount of binary or well structured data from semantic, more abstract data, two databases are used. The Model Database stores data of the first type and the Knowledge Graph stores semantic data.

Model Database: The content of the Model Database are learned features of the face recognition. If a person is learned by an entity, the Model Database is updated and the features are synchronized with all currently registered entities. This ensures that a person learned by a single entity can later be recognized by any other entity. Simultaneously, every added person is also inserted in the Knowledge Graph as new instance of a person concept (see next paragraph). After a successful insertion, the Knowledge Graph returns a UUID that is used for indexing the features in the Model Database. Hence, the Model Database has no information about properties of persons (e. g., their names), but just the learned features indexed by UUIDs. Note, the Knowledge Graph has no information about visual features of a person's face but it contains all semantic information. Both types of information are coupled by the UUID.

Knowledge Graph: The Knowledge Graph consists of two parts. The first part is Memory Net [21], which defines the graph structure and an API for accessing the structure. The second part is an underlying Neo4j graph database. The Memory Net API provides the following functions:

- insert/update information (e. g., measurements or states) obtained from entities
- retrieve information about the state of the environment
- provide basic reasoning (e. g., spatial and temporal reasoning)

The graph structure of a Memory Net consists of concepts with properties defined as a property graph [22]. Concepts are sub-graphs and properties are quantities in the physical world that describe a concept. Concepts and properties defined in this work are:

- concepts: room, object, person, passage, entity, capability, measurement, and event
- properties: shape, utterance, and position

Capability concepts are special, since they can be linked to an executable high level action of a robot. Names of persons or entities are described by utterance properties and their location is specified with the position property. A room is described by its shape, position, name (utterance) and its connections to other rooms (passages).

An entity can provide measurements of its environment like detected persons or its own position. Such measurements are updated continuously in the graph and they link the raw data, the observer (entity), and properties of any concept extracted from these measurements. Note, all nodes have timestamps indicating when they have been created or modified. A timestamp usually contains the time when a measurement has been acquired by a sensor. If no sensor timestamp exists, the time when the measurement is inserted into the knowledge graph is used.

VI. Planning and Re-planning

The main purpose of the system is to offer services to users by exploiting the collected observations and flexibly combining the available capabilities. For this we treat a service request as a planning problem that we represent in PDDL [23, 2.1 level 2] and that we solve using a standard symbolic planner. In general PDDL requires to define three aspects: (i) An initial state: This is a set of facts that is dynamically queried from the Knowledge Graph, including e. g. information about which rooms are connected, which entities are active, and in which room objects/persons/entities have been observed lately. (ii) A goal state: This is a set of desired facts, which is inferred from the interaction with the user, e. g. an object/information should be given to a person, or a person should be brought to another person or into a certain room. (iii) The actions: This is a set of rules that describe under which conditions a capability can be executed and what is the resulting change in state/facts. Each action is associated with a cost.

Based on this problem definition the Metric-FF [24] planner is used to find the cost-optimal sequence of actions that fulfill the goal state. For a goal state of `person_at paul library` a possible plan can be:

1) `entity_change_room johnny hallway kitchen`
   execute: move(where="kitchen") on "johnny"
2) `entity_invoke_person johnny paul kitchen`
   execute: inform(who="paul", what="Go to the library!") on "johnny"
3) `person_change_room paul kitchen library`
   assume: "paul" moves to "library"

In each step the first token is the action name which is directly mapped to a capability. The second token is the agent that should perform the capability, and the remaining tokens contain the required parameters of the capability. If the agent is an entity (step 1), the backend can directly ask it to execute the capability remotely, e. g. to move somewhere. If a human agent should perform an action (step 3), beforehand the backend has to use an entity to tell the human what to do (step 2). The given instruction (parameter *what*) is dynamically aggregated from the following actions that involve this person.

The costs for an action mainly depend on the involved agents. Entities should be preferred whenever possible. Therefore they are associated with a low cost. Persons have a cost that is between one and two orders of magnitude higher. The value increases with the age of the last observation of the person. This favors recently seen persons and thus increases the probability of a successful plan execution. Besides using the age of the last observation, currently no other means of representing uncertainty about the current state are considered.

An important aspect for acceptance of the system is that humans in the Living Lab can intuitively understand what the system and each entity is currently doing. When the backend receives a service request it uses the Smart Lobby to express the desired goal state, if a plan was found, and which agents are involved in the plan. During plan execution, for each step an explanation is shown and potential errors are expressed. Furthermore, each entity explains which capability it is about to perform if this is not evident inherently. For instance, before a robot starts moving it will tell to which room it wants to go and shows this information on its screen. All explanations are given in natural language, which requires to generate sentences from symbolic information and to have a dedicated error message policy.

A capability can fail due to a technical problem (e. g., a component crashes) or due to a wrongly assumed fact (e. g., a person is not in the room, where it was before). In both cases the plan will be aborted but the system can potentially find a new plan by selecting an alternative capability/entity or person. Currently we only consider the latter case. For instance, if a person is not found in a room this is stored as a special event in the Knowledge Graph. As a consequence the person is no longer assumed to be in that room, when the initial state is queried again.

The complexity of the planning problem heavily depends on the number of rooms, objects, agents and actions. The planner usually finds a solution within few seconds if it exists. After 10 seconds the planner is aborted, assuming that it is too time-consuming or impossible to find a cost-optimal solution. In such a case the planner could be restarted with parameters that put less focus on a cost-optimal solution. Though correct, this results in sometimes funny, but usually long and tedious plans. Thus this is not done in the running system.

VII. Experiments

In the experiments we demonstrate the flexibility of our system. We describe different settings of the environment whereas the goal state of the task stays the same.

For the experiments the area of the Smart Lobby and an office is used. Involved entities are the Smart Lobby and the mobile robot Johnny. As explained, persons are modelled in the system and can be invoked in the execution of plans.

The basic story is that Paul wants to have an object (the key) that is stored in the office. Therefore the goal state for all experiments is `person_has Paul key`. The speciality of the object is that only dedicated persons have access to the object storage. In our case this special person is Lisa.

Figure 3 shows the different environment settings. In all settings the Smart Lobby (☺) is available, the key's (🔑) location is in the office, and Paul's ($\mathcal{P}$) location is in the Smart Lobby. In setting (a) and (b), Johnny (☻) is available. In setting (a) and (d), Lisa's ($\mathcal{L}$) location is in the office, while in setting (b) and (c) she is located in the Smart Lobby.

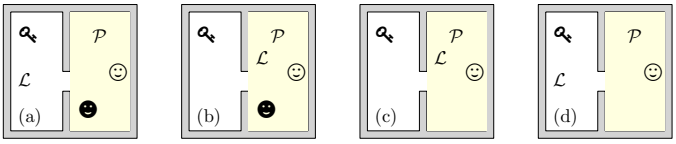

Fig. 3: Settings for the experiment. The position of Lisa and the availability of Johnny differ. The white area is the office and the other area is the Smart Lobby. $\mathcal{P}$ Paul; $\mathcal{L}$ Lisa; ☺ Smart Lobby; ☻ Johnny; 🔑 key

The system creates for each setting a plan which is shown in Fig. 4.1 to Fig. 4.4. If an entity is informing a person, there will be symbols (*/#) linking to the spoken text in Fig. 4.5.

0. Request control of entities
1. Backend invokes Johnny
2. Johnny moves to the office
3. Johnny invokes Lisa: ***
4. Lisa fetches the key
5. Lisa gives the key to Johnny: **
6. Johnny moves to the Smart Lobby
7. Johnny gives the key to Paul: *
8. Release control of entities

Fig. 4.1: Plan setting (a)

0. Request control of entities
1. Backend invokes Johnny
2. Johnny invokes Lisa: #
3. Lisa moves to the office
4. Lisa fetches the key
5. Johnny moves to the office
6. Lisa gives the key to Johnny: **
7. Johnny moves to the Smart Lobby
8. Johnny gives the key to Paul: *
9. Release control of entities

Fig. 4.2: Plan setting (b)

0. Request control of entities
1. Backend invokes Smart Lobby
2. Smart Lobby invokes Lisa ###
3. Lisa moves to the office
4. Lisa fetches the key
5. Lisa moves to the Smart Lobby
6. Lisa gives the key to Paul
7. Release control of entities

Fig. 4.3: Plan setting (c)

0. Request control of entities
1. Backend invokes Smart Lobby
2. Smart Lobby invokes Paul: ##
3. Paul moves to the office
4. Paul takes guidance of Lisa
5. Paul, Lisa move to the Smart Lobby
6. Paul releases guidance of Lisa
7. Smart Lobby invokes Lisa: ###
8. Lisa moves to the office
9. Lisa fetches the key
10. Lisa moves to the Smart Lobby
11. Lisa gives the key to Paul
12. Release control of entities

Fig. 4.4: Plan setting (d)

*** "Please fetch the key and give the key to me and press the acknowledge button!"
** "Please give the key to me and press the acknowledge button!"
* "Please take the key!"
# "Please go to the office and fetch the key and give the key to me and press the acknowledge button!""
"Please go to the office and bring Lisa to the Smart Lobby!"
"Please go to the office and fetch the key and go to the Smart Lobby and give the key to Paul!"

Fig. 4.5: Information texts

In setting (a) the plan (Fig. 4.1) is straight forward. The system sends Johnny into the office (asking Paul would lead to much higher costs for the plan) and lets it ask for the key. Since Lisa is assumed to be at the office, she is assumed to fetch the key and to give it to Johnny. After receiving the key (checked by the usage of the acknowledge button), Johnny returns to the Smart Lobby. Then the key is given to Paul, which satisfies the goal state.

In setting (b, Fig. 4.2) all agents are in the Smart Lobby. The system lets Johnny inform Lisa to move to the office for picking up the key. Johnny moves to the office too and waits for receiving the key from Lisa. Then, Johnny moves back to the Smart Lobby and gives the key to Paul which satisfies the goal state. The number of tasks for Lisa is kept to a minimum, because otherwise it would increase the costs of the plan significantly. An equal plan in terms of costs could also involve the Smart Lobby entity for informing Lisa.

In setting (c, Fig. 4.3) all persons are in the Smart Lobby and Johnny is unavailable. Since Lisa is the only person with access to the key she is directly asked by the Smart Lobby to move to the office, to fetch the key, and to bring it back to Paul. If this happens as expected the goal state is fulfilled.

In setting (d) the static Smart Lobby is the only available entity again and the system executes the plan of Fig. 4.4. Since Paul is the only person in reach for the system, he is asked to guide Lisa into the Smart Lobby. Therefore he needs to go into the office where Lisa is supposed to be and pick her up. This seems to be strange but the aim of the system to interact directly with Lisa in order to ask her for the key. So far we did not model any mechanism that allows persons to inform other persons about something which would be an alternative to the current plan. When Lisa enters the Smart Lobby she will be informed (###) to fetch the belt and to give it to Paul, which again satisfies the goal state.

VIII. Conclusion

In this paper we presented a system architecture for a multi-entity intelligent system. It mainly focuses on flexibility in different aspects to allow for a robust task achievement. First, the system allows for dynamic registration of previously unknown entities with different capabilities. Second, humans are considered as part of the system by assuming their capabilities and planning with them. Third, planning problems are dynamically generated based on which entities are registered, what capabilities they have, and the relevant state of the environment. As shown with the experiments in Section VII, these aspects together allow for achieving tasks in different situations.

As for all such systems as presented in this paper there are loose ends that allow for improvement in future work. This includes improving the planning method towards allowing for parallel execution of capabilities as well as considering uncertainty information. Furthermore, the usage of the semantic knowledge representation would allow for incorporating external world knowledge and more sophisticated reasoning processes. And finally, the system is to be evaluated during long-term operation in our *Living Lab*.

References

[1] S. Rebhan, N. Einecke, and J. Eggert, "Consistent modeling of functional dependencies along with world knowledge," in Proceedings of the International Conference on Cognitive Information Systems Engineering, 2009, pp. 341–348.

[2] S. Rebhan, A. Richter, and J. Eggert, "Demand-driven visual information acquisition," in Computer Vision Systems, 7th International Conference on Computer Vision Systems, ICVS, 2009, pp. 124–133.

[3] S. Harnad, "The symbol grounding problem," Physica D: Nonlinear Phenomena, vol. 42, no. 1, pp. 335–346, 1990.

[4] N. Nikolakis, N. Kousi, G. Michalos, and S. Makris, "Dynamic scheduling of shared human-robot manufacturing operations," Procedia CIRP, vol. 72, pp. 9–14, 2018.

[5] N. Nikolakis, K. Sipsas, P. Tsarouchi, and S. Makris, "On a shared human-robot task scheduling and online re-scheduling," Procedia CIRP, vol. 78, pp. 237–242, 2018.

[6] M. Krüger, C. B. Wiebel, and H. Wersing, "From tools towards cooperative assistants," in Proceedings of the 5th International Conference on Human Agent Interaction. ACM, 2017, pp. 287–294.

[7] M. K. Lee, J. Forlizzi, P. E. Rybski, F. Crabbe, W. Chung, J. Finkle, E. Glaser, and S. Kiesler, "The snackbot: documenting the design of a robot for long-term human-robot interaction," in Proceedings of the 4th ACM/IEEE international conference on Human robot interaction. ACM, 2009, pp. 7–14.

[8] S. Rosenthal, J. Biswas, and M. Veloso, "An effective personal mobile robot agent through symbiotic human-robot interaction," in International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), vol. 1, May 2010, pp. 915–922.

[9] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrová, J. Young, J. Wyatt, D. Hebesberger, T. Koertner, R. Ambrus, N. Bore, J. Folkesson, P. Jensfelt, L. Beyer, A. Hermans, B. Leibe, A. Aldoma, T. F., and M. Hanheide, "The STRANDS project: Long-term autonomy in everyday environments," IEEE Robotics & Automation Magazine, vol. PP, 04 2016.

[10] T. Chakraborti, S. Kambhampati, M. Scheutz, and Y. Zhang, "AI challenges in human-robot cognitive teaming," CoRR, vol. abs/1707.04775, 2017.

[11] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila, "Toward human-aware robot task planning." in AAAI spring symposium: to boldly go where no human-robot team has gone before, 2006, pp. 39–46.

[12] E. Tosello, Z. Fan, and E. Pagello, "A semantic knowledge base for cognitive robotics manipulation," University of Padova, 2016.

[13] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, and H. S. Koppula, "Robobrain: Large-scale knowledge engine for robots," CoRR, vol. abs/1412.0691, 2014.

[14] Y. Jiang, N. Walker, J. W. Hart, and P. Stone, "Open-world reasoning for service robots," in International Conference on Automated Planning and Scheduling,, 2019, pp. 725–733.

[15] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots," International Journal of Robotic Research, vol. 32-5, pp. 566–590, 2013.

[16] L. Fischer, S. Hasler, J. Deigmöller, T. Schnürer, M. Redert, U. Pluntke, K. Nagel, C. Senzel, J. Ploennigs, A. Richter, and J. Eggert, "Which tool to use? Grounded reasoning in everyday environments with assistant robots." in CogRob@ KR, 2018, pp. 3–10.

[17] M. Hanheide, M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, et al., "Robot task planning and explanation in open and uncertain worlds," Artificial Intelligence, vol. 247, pp. 119–150, 2017.

[18] S. Hasler, J. Kreger, and U. Bauer-Wersing, "Interactive incremental online learning of objects onboard of a cooperative autonomous mobile robot," in International Conference on Neural Information Processing. Springer, 2018, pp. 279–290.

[19] S. Fuchs, N. Einecke, and F. Eisele, "Smartlobby: Using a 24/7 remote head-eye-tracking for content personalization." in UbiComp, 2019, p. accepted.

[20] S. Fuchs, N. Einecke, M. Mühlig, B. Bolder, and F. Eisele, "Smartlobby: A 24/7 human-machine-interaction space within an office environment." in European Conference on Ambient Intelligence (AmI), 2019, p. accepted.

[21] J. Eggert, J. Deigmöller, L. Fischer, and A. Richter, "Memory Nets: A knowledge representation for autonomous entities." in International Conference on Knowledge Engineering and Ontology Development, 2019, p. accepted.

[22] R. Angles, "The property graph database model," in AMW, ser. CEUR Workshop Proceedings, vol. 2100. CEUR-WS.org, 2018.

[23] M. Fox and D. Long, "PDDL2. 1: An extension to PDDL for expressing temporal planning domains," Journal of artificial intelligence research, vol. 20, pp. 61–124, 2003.

[24] J. Hoffmann, "The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables," Journal of artificial intelligence research, vol. 20, pp. 291–341, 2003.