A Probabilistic Multi-Agent System Architecture for Reasoning in Smart Homes

Conference Paper · July 2019

DOI: 10.1109/INISTA.2019.8778306

Probabilistic Logic Reasoning in Multi-Agent Based Smart Home Environment

Dagmawi Neway Mekuria, Paolo Sernani, Nicola Falcionelli, Aldo Franco Dragoni

Abstract The dynamic nature of the ambient assisted living (AAL) environments and vague human communications may result in ambiguous, incomplete and inconsistent contextual information. These ultimately lead to uncertainty, which inevitable in smart home environments due to inaccurate sensor data or due to the existence of unobserved variables for privacy reasons. Aiming at tackling some of these challenges, this paper applies a probabilistic logic reasoning technique into multi-agent based smart home architecture. Accordingly, this study shows how the probabilistic reasoning technique enables the agents to reason under uncertainty. Further, it discusses how the intelligent agents enhance their decision-making process by exchanging information about missing data or unobservable variables using agent interaction protocols. Moreover, the paper presents the proof-of-concept implementation and preliminary experimental evaluation of the proposed smart home system. In general, the study demonstrates that the combination of multi-agent system (MAS) technologies and probabilistic logic programming can help in building a reasoning system, which is capable of performing well under vague inhabitant commands and missing information in partially observable environments.

Key words: Smart homes, Reasoning under uncertainty, Multi-agent systems

Dagmawi Neway Mekuria
Università Politecnica delle Marche, Ancona Italy, e-mail: d.n.mekuria@pm.univpm.it

Paolo Sernani
Università Politecnica delle Marche, Ancona Italy e-mail: p.sernani@univpm.it

Nicola Falcionelli
Università Politecnica delle Marche, Ancona Italy e-mail: n.falcionelli@pm.univpm.it

Aldo Franco Dragoni
Università Politecnica delle Marche, Ancona Italy e-mail: a.f.dragoni@univpm.it

1 Introduction

The term *"smart home"* refers to a residence equipped with technologies that facilitate monitoring of residents, promote independence and increase the quality of life [9]. Smart home systems have been widely applied to home energy management [1], health-care [11], and comfort centered services [18]. To effectively deliver these services, the smart home system needs to perceive the state of the residence through sensors, and automatically adapt the living environment to its inhabitants preferences through actuators. The automatic adaptation process of the living environment is mainly determined and controlled by the reasoning system, which is considered to be the brain of the smart home system. Precisely, the primary role of a smart home reasoning system (SHRS) is to make appropriate decisions towards achieving the comfort and efficiency goals of the inhabitant and their environment.

Recent investigations in SHRSs identified reasoning under uncertainty and incomplete knowledge as the major challenges of these systems. Specifically, [14] underlined that uncertainty is inevitable in ambient assisted living environments as sensors may read inaccurate data or due to the existence of unobserved variables for privacy reasons. Further, the dynamic nature of the environment and vague human communications may result in ambiguous, incomplete and inconsistent contextual information, which ultimately lead the system into uncertainty. However, little attention has been paid to address these challenges.

The present article aims to tackle some of these challenges by proposing a probabilistic multi-agent system architecture for reasoning in the AAL environments. For this, it utilizes the notion of intelligent agents and a probabilistic logic programming technique. On the basis of these technologies, the paper defines multiple intelligent agents and discusses their purposes and operations. Mainly, it illustrates how the agents perform reasoning under uncertain situations, using an ambiguous inhabitant command as a scenario. Finally, the feasibility of the proposed MAS architecture is demonstrated using a proof-of-concept (PoC) implementation.

The rest of the paper is organized as follows: Section 2 gives an overview of the related works. Section 3 briefly introduces the ProbLog language and agent interaction protocols. In section 4, we present our multi-agent system architecture and discusses the probabilistic reasoning process using a simple scenario. Section 5, evaluates the proof-of-concept implementation of the proposed system presenting the experimental settings and the preliminary test results. And, section 6 concludes the paper.

2 Related Research

Over the years, several studies have exploited the basic advantages of intelligent agents to model smart home environments and to automatically control their overall operations. For instance, [16] introduced a multi-agent system architecture to provide health care services in the AAL environment. The core of the system is a belief-desire-intention (BDI) agent, which represents the reasoning module of the overall architecture. Similarly, [15] proposed ThinkHome, a smart home architecture composed of a knowledge base and a multi-agent system. ThinkHome is populated by various specialized BDI agents, which are responsible for solving different problems by utilizing ontological reasoning methods. Further, [1] modeled the home environment as a MAS, and utilized contract based negotiation protocol for power management in home automation domain. Each of these contributions showed the advantages of modeling an AAL environment in terms of MAS, however, none of them consider issues related to uncertainty while presenting their reasoning modules. On the other hand, the MavHome architecture [6], which is a hierarchy of collaborative rational agents, was designed to meet the overall goals of a smart home environment. In MavHome each agent is composed of four cooperating layers: Decision, Information, Communication, and Physical. The decision layer is built by combining different machine learning algorithms, including Markov decision process. However, the contribution barely discusses issues related to uncertainty and methods to handle them in AAL environments.

Some other studies attempted to tackle uncertainty related challenges in the AAL environment using probabilistic graphical models. For instance, [13] utilized Multi-Entity Bayesian Networks to present a reference model for the AAL system that deals with uncertainty. However, the contribution is more tailored to the detection and prediction of unwanted situations than decision-making under uncertain conditions. Likewise, [21] discussed a smart home reasoning scenario which incorporates uncertainty reasoning using a rule-based system and Bayesian networks. And, [5] presented a framework to build a home automation system reactive to voice commands. In which, Markov logic network (MLN) is used to build the decision-making module of the system, and the uncertainty of the decision model was learned from data. Further, the authors integrated and tested their proposed system in a real-world environment, and presented some interesting results. However, both of the aforementioned works did not structure their solutions as a MAS, thus did not benefit from the advantage of autonomous agents. Further, MLNs are known to require a non-trivial effort by experts to properly model uncertainties in terms of weights [4]. Taking this into account, [20] utilized ProbLog for complex activity recognition in the smart home domain but again did not discuss a reasoning system as a whole.

Having these in mind, the present paper utilized the combination of MAS and probabilistic logic programming techniques to tackle uncertainty related issues in AAL environments. Specifically, in this paper, the smart home system is modeled in terms of collaborative intelligent agents, and probabilistic reasoning is utilized to give the agents an ability to make a decision under an uncertain situation.

3 Background

This section briefly introduces the probabilistic logic programming technique and the agent interaction protocols utilized in this paper.

3.1 ProbLog

ProbLog is a probabilistic extension of Prolog, and like Prolog, its program consists of a set of definite clauses. However, in ProbLog every clause c_i is annotated with the probability p_i that it is true, and these probabilities are mutually independent with each other [7]. That is $P(A \cap B) = P(A)P(B)$. Listing 1 shows a simple ProbLog program [1], in which the first clause indicates the fact burglary is true with probability 0.3 and false with probability 0.7. And the second clause indicates, if burglary is true, then alarm will be true as well with 0.9 probability.

```
0.3::burglary.
0.9::alarm:-burglary
```

Listing 1: Simple ProbLog program

Unlike Prolog, where one is interested in determining whether a query succeeds or fails, in ProbLog we are interested in computing the probability that it succeeds. Below, the equations for computing the success probability of a ProbLog query are cited and summarized from [7] and [20].

Given a ProbLog program $T = \{p_1 : c_1, ..., p_n : c_n\}$. The probability of a world ω, that is a certain instance of the program, is defined as follows:

$$P(\omega|T) = \prod_{c_i \in \omega} p_i \prod_{c_i \in \omega_T \setminus \omega} (1 - p_i) \tag{1}$$

Where $\omega_T \setminus \omega$ describes the set of clauses that were not instanced in ω but are part of T, i.e. the set of false ground probabilistic atoms. Then, the success probability of a query q in the ProbLog program T is computed as follow:

$$P(q|\omega) = \{ 1 , \exists\theta : \omega \models q\theta 0, otherwise \tag{2}$$

$$P(q, \omega|T) = P(q|\omega).P(\omega|T) \tag{3}$$

$$P(q|T) = \sum_{\omega \subseteq W} P(q, \omega|T) \tag{4}$$

In short, the probability that a ProbLog query q succeeds is the sum of the probabilities of those worlds where q can succeed.

[1] This program is a shortened form of an example provided in the official website of ProbLog.

3.2 Agent Interaction Protocols

In a MAS environment, it is essential to have a standard agent interaction mechanism, that allows agents to collaborate and coordinate with each other in order to achieve their goals. Hence in our proposed MAS architecture, we utilized the FIPA Request Interaction Protocol (IP) [12] and Contract Net Protocol (CNET) [19]. IP allows one agent, the *Initiator*, to request another agent, the *Participant*, to perform an action. The Participant processes the request and makes a decision whether to *accept* or *refuse* the request. Once the request has been agreed upon, the Participant must communicate a *failure*, an *inform-done* or *inform-result* messages for the initiator agent [3]. Likewise, CNET is a market-based protocol which allows the Manager agent to have some task performed by one or more other Contractor agents. In both IP and CNET, at any time an agent can be an Initiator/Manager, a Participant/Contractor or both.

4 The Multi-Agent System Architecture

Home is a place where one lives. Commonly, it is composed of several rooms, dedicated for the specific activities of daily living. As shown in figure 1, our proposed multi-agent system architecture follows this logical structure of the home environment. Specifically, a group of collaborative intelligent agents manages each room of the smart home. These agents specialize in the specific tasks of the home automation system (such as monitoring the brightness or the air conditioning of the room) and collectively control the comfort and efficient operation of that specific room. In addition, these agents collaborate with other agents, which resides in another room of the home, to achieve the overall goals of the smart home system. In order to realize this architecture, we propose four kinds of room-level agents: *Device, Service, Reasoner, and Negotiator* agents. In general, this kind of architecture allows to benefit from the basic advantages of intelligent agents such as autonomy, social ability, reactivity, and pro-activeness. It enables to modularize the smart home system into autonomous, collaborative and distributed components, which provide well-tailored services based on their location in the home. Further, the architecture allows to design a highly customizable and fault tolerant smart home system. Below the purposes and functioning of the aforementioned agents are discussed.

4.1 Device agents

In the proposed architecture, a device agent (DA) controls the overall operation of a smart home device (i.e a sensor or an actuator) or group of devices. A DA which is responsible for controlling a specific sensing device(s), continuously monitors the changes in the environment and combines low-level sensor readings with other

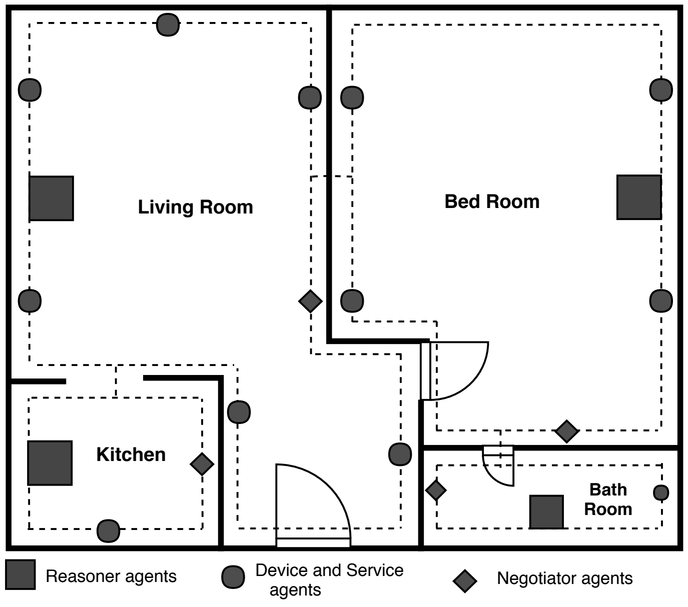

Fig. 1: The smart home floor plan

data in the home to generate high-level contextual information about the state of the home or its inhabitant. For instance, a device agent which controls room temperature, monitors a group of temperature sensors in that room and generates contextually meaningful information (e.g. *warm, cold ...*) by processing the sensor data with the season of the year, time of the day, type of the room etc. In addition, device agents are also able to determine their degree of belief (*Bel*) for the generated contextual information (e.g. *Bel(warm) = 0.9*). Here, a degree of belief can be understood as a number in the range of $[0,1]$, that represents the measure of the agent's confidence in the generated context based on all available evidence. In order to combine separate pieces of information and determine its *Bel* about the newly derived context, a DA may use evidential theories such as Dempster-Shafer theorem [17] or other probabilistic approaches.

On the other hand, a device agent which controls the operation of an actuator(s) put into consideration the current situation of the home and its inhabitants, while executing a user command or during a self-adaptation process. For instance, when operating an electrical device the agent should check the real-time electric prices, and schedule the operation for off-peak hours if the outcome of the operation is not urgently required. This ensures the efficient execution of the command on the targeted smart home device and renders high-quality services in a cost effective way possible.

4.2 Service agents

Service agents (SA) are general purpose agents which provide house level information, that is not specific to a single room or space in the home environment. Global information, which can be acquired from external data sources (such as weather data, and real-time electricity price) and information from other smart home software components (such as activity recognition and person localization systems) can be coupled into services agent and integrated into the proposed multi-agent system architecture. Like device agents, SAs are also able to determine and share their degree of belief about their produced information.

4.3 Reasoner agents

In the proposed MAS architecture, each room is equipped with a reasoner agent (RA), which is responsible for the automatic control of the room environment and its adaptation to the inhabitant's needs. The decision-making unit of this agent is designed based on a probabilistic logic programming technique called ProbLog. This technique allows the design of a hybrid reasoning system, which benefits from the advantages of both symbolic and statistical artificial intelligence methods. As a result, along with other assets of hybrid systems, RA possesses the ability to act under uncertainty and perform well with erroneous sensor data and ambiguous user commands. In addition, ProbLog enables to learn the structure and parameters of the probabilistic rules from data and to model the system using human-readable and easily modifiable rules. Besides, these agents are designed to cope with unavailable, conflicting and inconsistent sensor data by actively collaborating with other agents in the system.

Algorithm 1 details the overall operation of an RA when it receives a user command or detects a change in the home. When an RA receives a user command, first it determines the smart home service associated with it (e.g. the user command "*turn on the light*" can be associated with the smart home service "*light*"). Following this, it discovers a list of device and service agents which provide relevant information to the requested service, and then it identifies the device agent which controls the behavior of that specific smart home service (lines 2-3). Subsequently, it will send an information request about the current state of the inhabitant and its environment to the data provider agents (line 4). Whereas, when a DA receives an information request, it determines the value of the requested information and its degree of belief about it (e.g. *useractivity="cooking", Bel=0.99*), and send back a reply for the reasoner agent. However, some of the data provider agents may fail to provide the requested information on time or may fail to reply at all due to sensor failure or other problems. Moreover, RA may discover some inconsistencies or conflicts in the collected data (lines 5-6). In this kind of situations, RA will request its negotiator agent to interact with other negotiator agents in the system and gather the missing data from similar data provider agents in other rooms of the smart home (line 7). When the interaction process ends, the negotiator agent will transfer the result to the reasoner agent, which will use the data to improve its level of uncertainty or resolve conflicts, so that it makes the most appropriate decision (lines 8-9). When RA receives all information, it will check again if some data are still missing (lines 10-11), if so, it will use default values for these data (lines 12-13). Here, the default values of a sensor can be determined from the values of other sensors in the environment, and the current contexts of the home. After resolving the missing information, RA will build the ProbLog model and checks if it has the ability to run it locally (lines 16-17). RA's ability to solve a ProbLog model is determined by the local availability of a ProbLog engine and computational resources required to solve the model. If RA has reasoning ability, it solves the ProbLog model locally (line 19), that is determining the success probabilities of all the associated commands related to the user request and the situation of the home. Afterwards, it picks the command with the highest probability, that is the command considered to give maximum user satisfaction in the current state of the home. Whereas, if RA does not have reasoning ability, it will request the negotiator agent in the same room to delegate the reasoning task for other RAs in the system (line 21). If the negotiator agent fails to find another RA willing to solve the model, RA will take the default command for the user request (line 26). Otherwise, it will use the result of the interaction, which is a command with the highest success probability to serve the user request (line 24). Finally, RA will send the control command for the devices controller agent (line 29). Section 4.3.1 briefly describes the underlying decision-making process of this agent, using a simple scenario, and figure 2 depicts the sequence diagram of the agents interaction in the scenario. The function and operation of the negotiator agent are discussed in the next section.

4.3.1 Scenario: Dealing with ambiguous user command

Suppose the kitchen has four kinds of lights (*i.e. ceiling, sink, dining and cooking lights*) and the inhabitant stands near the kitchen-sink (*loc="near-sink", Bel=0.95*) while preparing her dinner (*act="cooking", Bel=0.95*). Meanwhile, she issues a voice command *"turn on the light"* to the system, without specifying the exact light she wants the system to turn on. This kind of ambiguous user commands are one of the major sources of uncertainty in smart home environments, and hereinbelow we will use this scenario to illustrate how the reasoner agent function under such circumstances.

To simplify the discussion let us again assume that, RA controls the lighting of the room only based on the inhabitant's current location and activity. That is, the system turns on the *cooking, sink, dining or ceiling light*, when the inhabitant is around the kitchen cabinet and preparing food, around the sink and washing plates, on the dining table and eating, or anywhere in the room and tidying up respectively. The current location and activity of the inhabitant are also separately utilized to control the lighting of the room, with distinct probability values (*Pr*). That is, activity-based light control (*Pr=0.7*) is assumed to give better user satisfaction compared with

Algorithm 1: RA's reasoning algorithm

```
input : userRequest
$service \longleftarrow determineRequestedService(userRequest)$;
$dataProviderAgents \longleftarrow discoverDataProviders(service)$;
$devicesControllerAgent \longleftarrow discoverDevicesController(service)$;
$sensorData \longleftarrow collectSensorData(dataProvidersList)$;
$missingData \longleftarrow getMissingData(service, sensorData)$ ;
if not empty missingData then
    $sendInformationRequest(negotiator, missingData)$;
    $missingData \longleftarrow receiveData(negotiator)$;
    $sensorData.merge(missingData)$;
    $missingData \longleftarrow getMissingData(service, sensorData)$ ;
    if not empty missingData then
        $missingData \longleftarrow getDefaultValues(missingData)$;
        $sensorData.merge(missingData)$;
    end
end
$probLogModel \longleftarrow buildProbLogModel(sensorData, service)$;
$ableToReason \longleftarrow ableToReasonLocally()$;
if ableToReason then
    $controlCMD \longleftarrow solveProbLogModel(probLogModel)$;
else
    $sendReasoningTaskRequest(negotiator, probLogModel)$;
    $result \longleftarrow receiveReasoningResult(negotiator)$;
    if result.status = SUCCESS then
        $controlCMD \longleftarrow result.data$ ;
    else
        $controlCMD \longleftarrow getDefaultCommand(service, sensorData)$;
    end
end
$sendCommand(devicesControllerAgent, controlCommand)$;
```

location-based control (*Pr=0.5*). Further, for any *"turn on the light"* command, it is also assumed to be fair, if the RA turns on the ceiling light with *Pr = 0.5*. Listing 2, shows the ProbLog representation of the aforementioned scenario. The parameters of the probabilistic clauses can be mined from a dataset using a probabilistic rule learner tool such as ProbFOIL[8], or manually designed by a knowledge engineer. The parameters of the rules and facts in listing 2 are from the assumptions made in this scenario.

In light of the above scenario, when RA receives the voice command, it first determines the set of possible worlds based on the current activity and location of the inhabitant. The possible worlds are the list of commands, whose logical formulas evaluated as true. And, as shown in figure 3, *cooking light, sink light and dining light* commands are all each in one possible world, which is represented as solid line rectangles in the figure. Subsequently, the agent determines the success probabilities of each possible worlds using equation 1, and sum up the probability values of each command using equation 4. Finally, it executes the command with a larger total sum probability value (i.e. *"cookingLight"*, with *0.665* success probability). For

```
0.95::loc(near_sink).
0.95::act(cooking).
0.90::cookingLight:- act(cooking), loc(cooking_area).
0.70::cookingLight:- act(cooking).
0.50::cookingLight:- loc(cooking_area).
0.90::sinkLight:- act(washing), loc(near_sink).
0.70::sinkLight:- act(washing).
0.50::sinkLight:- loc(near_sink).
0.90::dinningLight:- act(dinning), loc(dinning_table).
0.70::dinningLight:- act(dinning).
0.50::dinningLight:- loc(dinning_table).
0.90::ceilingLight:- act(tidyingup).
0.50::ceilingLight.
```

Listing 2: RA's Light Control Rules

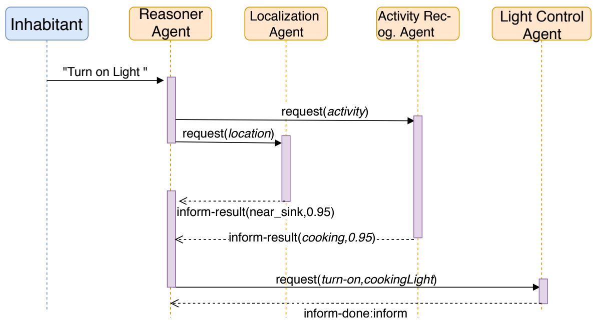

Fig. 2: Sequence diagram representing the interactions between the agents

the process of determining the possible worlds and their success probabilities, the reasoner agent fully relies on the ProbLog solver.

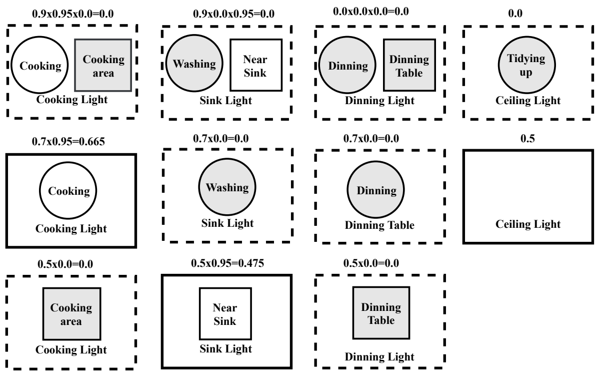

Fig. 3: All worlds for the turn on the light command

4.4 Negotiator agents

In the proposed MAS based smart home architecture, each room has a negotiator agent (NA), which is in charge of handling the collaboration process between agents of different rooms. Specifically, NA enables other agents in the system to exchange information about the state of the environment they are operating in. Further, when reasoner agents are not able to solve their decision-making problem due to computational resource limitation, the negotiator agent enables them to share their reasoning tasks (i.e. determining the success probability of a query) with other reasoner agents in the system. In practice, this agent is designed based on the FIPA Request Interaction and Contract Net Protocols, thus at different times, it acts as the *initiator* or *participant* of an information exchange process, or it acts as the *manager* or *contractor* of a negotiation process over reasoning tasks. As the primary focus of this paper is tackling some of the major causes of uncertainty in the smart home environments: that are *missing information, partial observability, and ambiguous user commands*, this section only discusses the information exchange processes between the intelligent agents, and leaves the discussion about the negotiation process over the distribution of reasoning tasks for future works.

NA plays the initiator role when it receives an information request message from the reasoner agent in the same room. As the Request Interaction Protocol initiator, the negotiator agent first looks for other negotiator agents in the house, who are registered to provide the information that the RA requested. Afterward, it will send a *request* message with the submission deadline, for all the information provider agents. Upon receiving the information from each of the participants (i.e. *inform-result* message), it will determine the best reply based on some criteria or combines separate replies to derive one, and communicates the final result for the reasoner agent. For instance, if RA is interested in the total number of people in the house after each participant NA communicated the number of people in their respective rooms, the initiator NA will sum the values, or if the requested information is room temperature, RA will determine the statistical mean value.

Likewise, NA plays the participant role, whenever it receives an information request message from another agent of the same type. Accordingly, based on the availability of a device agent in the same room, that is able to provide the requested information, the negotiator agent will accept or reject the request by sending *agree* or *refuse* messages respectively. If it accepts, it will send a *request* message for the information provider device agents. At last, when it receives the data from all device agents, it will process and communicate the result for the initiator NA. This process should help the reasoner agents to cope with sensor failures and perform well in partially observable environments.

5 Preliminary Evaluation

A Proof-of-concept implementation of the proposed multi-agent system architecture is done using the Java Agent DEvelopment Environment (JADE) framework [3] and ProbLog Python library. JADE is an open-source FIPA compliant middleware for developing multi-agent systems. It provides a ready-to-use and easy-to-customize MAS development platform, efficient agent communication mechanisms, effective agent life-cycle management, support for agent mobility, yellow and white page services, and a GUI for debugging and monitoring MAS application [3]. ProbLog is available as an online tool on the web and for download. The offline version offers both command line access to inference and learning and a Python library for building statistical relational learning applications [10]. For the purposes of giving the probabilistic reasoning ability for the reasoner agents in our system, we integrated the ProbLog Python library into the JADE MAS platform.

5.1 Experimental Setup

For the PoC implementation, we simulated the four-room smart home architecture presented in figure 1. In which, agents of each room run inside a JADE container hosted on a single board computer and connected with each other through a local area network. The MAS hosted in each node is composed of one Reasoner, one Negotiator, one JADE Directory Facilitator, and four Device and Service agents, that are Inhabitant Activity Recognition, Inhabitant Localization, Luminosity sensor, and Light service controller agents. The device and service agents are designed to randomly generates synthetic sensor data from their respective predefined set of values. For the purposes of heterogeneity, we host two of the rooms in Raspberry PI 3 Model B+ nodes (quad-core A53 (ARMv8) 64-bit @ 1.4GHz, 1GB SDRAM, Gigabit Ethernet, Raspbian OS), and the other two in Intel Galileo boards (single core i586 CPU @ 400 MHz, 256 MB DRAM, 100 Mb Ethernet, Yokto Linux OS). As the latest version of Yokto does not support Python 3, which is a prerequisite to run a ProbLog engine, the two rooms which are hosted in the Galileo boards were without the ability to solve ProbLog model, thus need to delegate their reasoning tasks to other reasoner agents in the system. In general, this setup allows us to measure the CPU time needed by the RA to reason locally on the Raspberry PI nodes, and the CPU time needed by the RA to negotiate and solve the ProbLog model on another node in the system. Further, to make a comparison of the reasoning time in the two hardware configurations, the latter case was tested both on the Raspberry Pi (with no local reasoning ability) and Galileo boards. In this evaluation, scenario one and its ProbLog model(presented in figure 2) is utilized. Figure 4 depicts the experimental setup used to run the tests and evaluate the proposed multi-agent system.

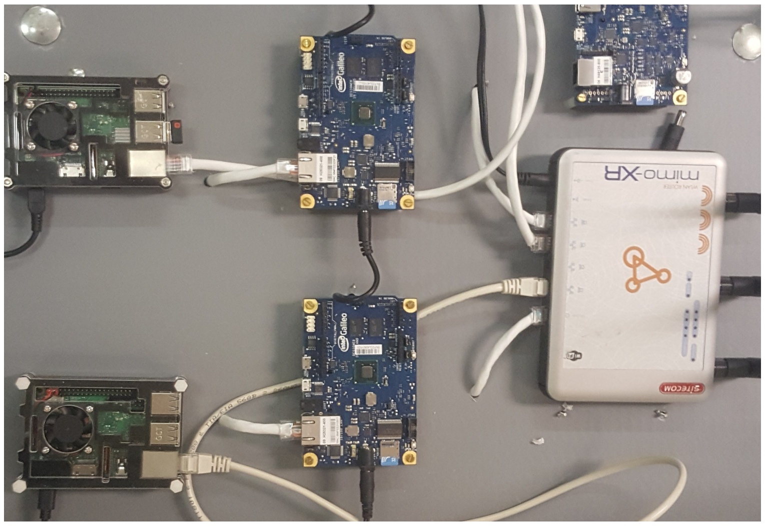

Fig. 4: The experimental setup.

5.2 Tests and Results

To measure the CPU time that an RA requires to reason, we run two groups of experiments. First, we measure the CPU time that an RA requires to perform the entire reasoning process locally on a Raspberry PI node. Second, we measure the CPU time that an RA requires to delegate and solve the ProbLog model via the negotiator agent, both on the Raspberry PI and Galileo nodes. Further, we conducted four distinct tests for each of the aforementioned two experiments, by incrementally changing the number of locally unavailable (missing) information required to reason from zero to three. We run all the tests 100 times and recorded the mean and standard deviation of the reasoning time. Table 1 and 2 summarize the results of the experiments, and figure 5, 6 and 7 present comparable results of each run.

Table 1: CPU time(ms) of reasoning locally vs. over negotiation with varying missing information(MI), on Raspberry PI 3 nodes.

Raspberry PI	No. of missing information			
	None	One	Two	Three
Reasoning locally	597.2±17	617.9±17	618.2±23	650.2±17
Reasoning over negotiation	674.9±28	727.6±34	716.13±32	729±28

As can be seen in table 1 and figure 6, the reasoner agent which run on the Raspberry PI node required a relatively small amount of time to reason locally compared with the one which needs to delegate the reasoning task to another reasoner agent.

Table 2: CPU time(ms) of reasoning via negotiation with varying missing information(MI), on Raspberry PI 3 vs. Galileo nodes.

Board	No. of missing information			
	None	One	Two	Three
Raspberry	674.9±28	727.6±34	716.13±32	729±28
Galileo	828±60	920±71	925.3±42	900.5±41

This is due to the negotiation process and the message exchange between the agents involved in it. Whereas, as shown in figure 7, the difference in reasoning time between the Galileo and Raspberry PI nodes is relatively big. Intuitively, this is due to the difference in the computational capacity of the two boards. In addition, in this experiment, we also noticed that the time required to collect missing information(MI) from other device agents in the system is nearly constant.

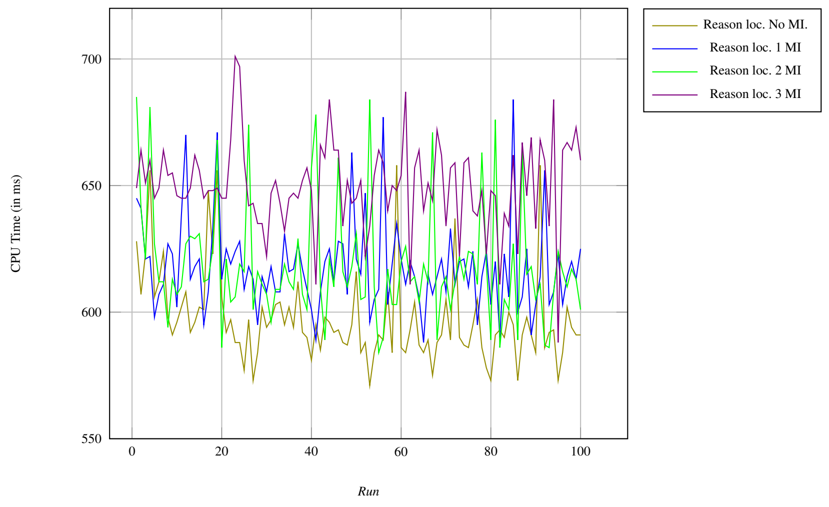

Fig. 5: Reasoning locally on the Raspberry PI nodes

In general, from the proof-of-concept implementation and preliminary experiments, we observed the practicality and feasibility of the proposed MAS based smart home architecture. First, it was observed that the probabilistic inference technique allows the reasoner agent to reason with ambiguous user commands and partial information. Second, collaborative MAS architecture enables the agents to cope well in partially observable environments. In addition, the MAS architecture makes it possible to design heterogeneous, highly customizable, robust and modular system.

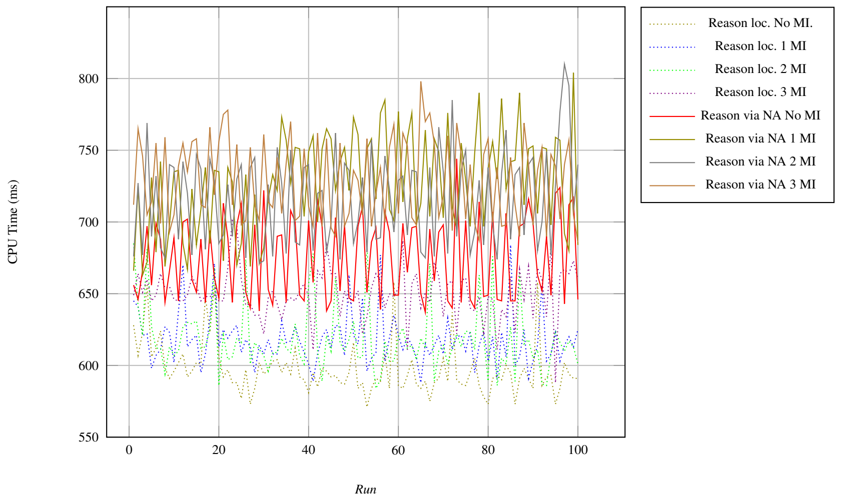

Fig. 6: Graphical comparison of reasoning locally and over negotiation on Raspberry PI nodes

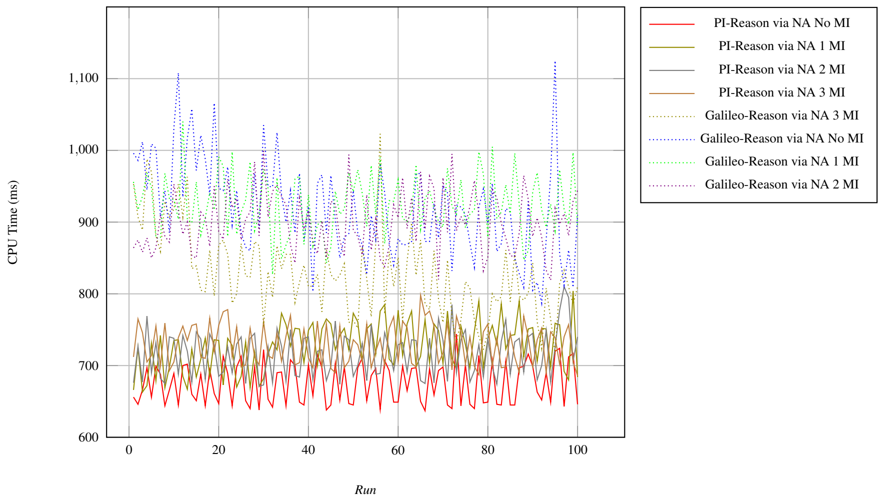

Fig. 7: Graphical comparison of reasoning over negotiation on Raspberry PI and Galileo nodes

5.3 *Threats to the validity of the experiments*

Being a preliminary study on the application of probabilistic logic reasoning in multi-agent based smart home system, the proposed experiments do inevitably suffer from threats to validity. Therefore, future works will address the identified limitations.

First, the core ProbLog model used in the experiments is based on the scenario discussed in section 3, so a model with few probabilistic logic clauses is not a computational challenge both for the proposed system and the integrated ProbLog solver. Second, all the device and service agents used in our PoC implementation are simulated, consequently, the data generation time of these agents was almost negligible. Therefore, to draw a strong conclusion about the reasoning time of the agents a more complex ProbLog model and real sensors data need to be used.

6 Conclusions and Future works

Prior works widely applied multi-agent systems to model AAL environments. However, most of them have not considered issues related to uncertainty reasoning, while presenting their decision-making components. Contrarily, few others tackled these challenges using probabilistic graph models, but either they have not discussed SHRSs as a whole, or they have not utilized the advantages of MAS to support the decision-making process in the AAL environments. With this in mind, in this study, we proposed a probabilistic multi-agent system architecture for reasoning in smart homes, based on a probabilistic logic programming technique called ProbLog and multi-agent system technologies. Accordingly, we illustrated how the probabilistic reasoning technique enables the intelligent agents to reason under uncertain situations. Further, we discussed how the agent interaction protocols enhance the decision-making process by allowing the agents to exchange information about missing data or unobservable variables between the agents. Therefore, this study showed that the integration of MAS and probabilistic logic programming can help in building a reasoning system for AAL environment, which is capable of performing well under vague inhabitant commands, missing information, and in partially observable environment.

Most notably, this is the first study to our knowledge which integrates the ProbLog reasoning engine into a multi-agent system framework and, to utilize it for tackling uncertainty issues in smart home environments. In general, our PoC implementation and experimental analysis suggest that this approach appears to be effective in counteracting uncertainties in the reasoning systems of these complex and dynamic environments. However, some limitations are worth noting. First, the proposed system is a work in progress, it is implemented as a PoC implementation, but not evaluated exhaustively. Second, some operations of the proposed system, especially the service and device agents are presented based on several assumptions. Third, the structure and parameters of the probabilistic logic rules are designed based on the subjective knowledge of the authors, yet can be learned from data. Finally, the application and advantage of CNET protocol for the distribution of the reasoning task are not discussed in detail. Therefore, future work will address these limitations and evaluate the proposed system thoroughly.

References

1. Abras, Shadi and Ploix, Stéphane and Pesty, Sylvie and Jacomino, Mireille: A multi-agent home automation system for power management. Informatics in Control Automation and Robotics, pp. 59–68. Springer (2008).
2. title=Steps toward artificial intelligence, author=Minsky, Marvin, journal=Proceedings of the IRE, volume=49, number=1, pages=8–30, year=1961, publisher=IEEE
3. Bellifemine, Fabio and Bergenti, Federico and Caire, Giovanni and Poggi, Agostino : JADE-a java agent development framework. Multi-Agent Programming, pp. 125–147. Springer, PLACE, (2005).
4. Bruynooghe, Maurice and Mantadelis, Theofrastos and Kimmig, Angelika and Gutmann, Bernd and Vennekens, Joost and Janssens, Gerda and De Raedt, Luc : ProbLog Technology for Inference in a Probabilistic First Order Logic.. ECAI, pp. 719–724. ACM, (2010).
5. Chahuara, Pedro and Portet, François and Vacher, Michel : Context-aware decision making under uncertainty for voice-based control of smart home. Expert Systems with Applications. **75**, pp. 63–79. (2017).
6. Cook, Diane J and Youngblood, Michael and Das, Sajal K: A multi-agent approach to controlling a smart environment. Designing smart homes, pp. 165–182. Springer (2006).
7. De Raedt, Luc and Kimmig, Angelika and Toivonen, Hannu : ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. IJCAI, pp. 2462–2467. Hyderabad, (2007).
8. De Raedt, Luc and Thon, Ingo : Probabilistic rule learning. International Conference on Inductive Logic Programming, pp. 47–58. Springer, PLACE, (2010).
9. Demiris, George and Hensel, Brian K : Technologies for an aging society: a systematic review of smart home applications. Yearbook of medical informatics. **17**, pp. 33–40. (2008).
10. Dries, Anton and Kimmig, Angelika and Meert, Wannes and Renkens, Joris and Van den Broeck, Guy and Vlasselaer, Jonas and De Raedt, Luc : ProbLog2: Probabilistic logic programming. Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 312–315. Springer, (2015).
11. Falcionelli, Nicola and Sernani, Paolo and Brugués, Albert and Mekuria, Dagmawi Neway and Calvaresi, Davide and Schumacher, Michael and Dragoni, Aldo Franco and Bromuri, Stefano: Indexing the Event Calculus: Towards practical human-readable Personal Health Systems. Artificial intelligence in medicine, (2018)
12. FIPA Request Interaction Protocol Specification. (2002)
http://fipa.org/specs/fipa00026/SC00026H.html Cited 10 May 2019
13. Machado, Alencar and Maran, Vinícius and Augustin, Iara and Lima, João Carlos and Wives, Leandro Krug and de Oliveira, José Palazzo Moreira : Reasoning on Uncertainty in Smart Environments. Proceedings of the 18th International Conference on Enterprise Information Systems, pp. 240–250, SCITEPRESS-Science and Technology Publications, Lda, (2016)
14. Mekuria, Dagmawi Neway and Sernani, Paolo and Falcionelli, Nicola and Dragoni, Aldo Franco : Reasoning in Multi-agent Based Smart Homes: A Systematic Literature Review. Italian Forum of Ambient Assisted Living, pp. 161–179. Springer, (2018).
15. Reinisch, Christian and Kofler, Mario J and Iglesias, Félix and Kastner, Wolfgang: Thinkhome energy efficiency in future smart homes. EURASIP Journal on Embedded Systems. **2011**, 1 (2011)
16. Sernani, Paolo and Claudi, Andrea and Palazzo, Luca and Dolcini, Gianluca and Dragoni, Aldo Franco : Home care expert systems for ambient assisted living: A multi-agent approach. Proceedings of the Workshop on The Challenge of Ageing Society: Technological Roles and Opportunities for Artificial Intelligence, Turin, Italy, (2003).
17. Shafer, Glenn : A mathematical theory of evidence. Princeton university press, (1976).
18. Si, Hua and Kawahara, Yoshihiro and Morikawa, Hiroyuki and Aoyama, Tomonori: A stochastic approach for creating context-aware services based on context histories in smart home. COGNITIVE SCIENCE RESEARCH PAPER-UNIVERSITY OF SUSSEX CSRP. **577**, 37 (2005)

19. Smith, Reid G : The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on computers. **12**, pp. 1104–1113. (1980).
20. Sztyler, Timo and Civitarese, Gabriele and Stuckenschmidt, Heiner : Modeling and reasoning with ProbLog: an application in recognizing complex activities. 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 259–264. IEEE, (2018).
21. Xiao, Biyi and Chen, Leiting and Liu, Ming and Cao, Yue and Yang, Yong : Design and implementation of rule-based uncertainty reasoning in Smart House. 2015 IEEE 16th International Conference on Communication Technology (ICCT), pp. 441–448. IEEE, (2015).