Contextual integration of heterogeneous data in an open and opportunistic smart environment : application to humanoid robots

Nathan Ramoly

université PARIS-SACLAY

NNT : 2018SACLL003

Thèse de doctorat

TELECOM SudParis

Contextual Integration of Heterogeneous Data in an Open and Opportunistic Smart Environment: Application to Humanoid Robots

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom SudParis

École doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Evry, 2 Juillet 2018, par

Nathan Ramoly

Composition du Jury :

Patrick Reignier Professeur, Grenoble INP (LIG UMR 5217)	Président
Dimitris Kotzinos Professeur, Université de Cergy-Pontoise (ETIS UMR 8051)	Rapporteur
Claudia Roncancio Professeure, Grenoble INP (LIG UMR 5217)	Rapporteur
Sao Mai Nguyen Maître de Conférences, IMT Atlantique (LAB-STICC UMR 6285)	Examinateur
Adriana Tapus Professeure, ENSTA-ParisTech (U2IS)	Examinateur
Amel Bouzeghoub Professeure, Télécom SudParis (SAMOVAR UMR 5157)	Directeur de thèse
Béatrice Finance Maître de Conférences HDR, UVSQ (DAVID)	Co-Directeur de thèse

Abstract

Personal robots associated with ambient intelligence are an upcoming solution for domestic care. In fact, helped with devices dispatched in the environment, robots could provide a better care to users. Although numerous works were conducted in that direction, major challenges are remaining. In this thesis, we aim to cover the problems of perception, cognition and action, in order to ensure the quality of service provided by robots in smart environments.

Although perception phase, consisting of context acquisition, is eased by using both sensors from robots and smart environments, such a variety of sources also brings issues of data quality and conflicts. Consequently, data and information provided by sensors can be imprecise, inaccurate, outdated, in contradiction, or simply missing. In other words, handling uncertainty is an essential challenge to tackle and leads to the following question: How can the robot acquire and understand the context while supporting all uncertainty issues? After perceiving, we enter in the cognition phase to reason and make a decision. The objective is to understand and identify anomalous situations that require the reaction of the robot. Consequently, the main issue is: how to detect these situations at runtime over uncertain data and knowledge in order to set a goal for the robot? Performing actions usually relies on task planning to reach a goal. As the smart home in which the robot will evolve is not known in advance, it is impossible to provide an exhaustive knowledge for task planning. Moreover, unexpected changes in the environment may occur during the plan execution. These problems lead to numerous task failures. Hence, task planning is facing the following challenge: How to understand the causes of failures in order to proactively avoid them in the future?

In this thesis, we proposed multiple contributions, exploring both reasoning and learning approaches. To tackle uncertainty, we proposed a novel context acquisition approach that deals with four uncertainty dimensions by semantically enriching and fuzzifying events, while existing techniques only cover two or three dimensions. In order to identify anomalous situations, we defined an hybrid solution based on Markov Logic Networks and a context ontology. By adding more semantic in the inference, we reach a more accurate situation identification even with partial observations. Finally task failures due to incomplete knowledge are tackled by a reinforcement learning method which identifies the causes. The knowledge of the task planner is then enriched with these detected causes. In dynamic environments, classical planners wait for the task failure to regenerate the whole plan and may have wasted time and energy to execute useless tasks. To avoid this, we propose to incrementally generate the plan and execute it on the fly in order to take into account the last context changes. Consequently, by combining these two proposals, task failures will be prevented in the future. As a result, all these contributions form a global framework that can be specified and configured for various robots and smart environments.

Each of our contributions was prototyped, tested and validated through simulations and/or physical tests using a small humanoid robot and a smart home development platform.

Résumé

L'association de robots personnels et d'intelligences ambiantes est une nouvelle voie pour l'aide à domicile. En effet, en s'appuyant sur les appareils intelligents présents dans l'environnement, les robots pourraient fournir un service de haute qualité. Cependant, même si plusieurs travaux vont dans cette direction, il y a encore de nombreux verrous. C'est pour cela que dans cette thèse, nous cherchons à résoudre les problématiques de perception, de cognition et d'action rencontrées par les robots dans des environnements intelligents, et ainsi assurer la qualité de service de ces derniers.

Même si la perception est facilitée par l'utilisation des capteurs à la fois de l'environnement intelligent et du robot , une telle variété de capteurs est une source de conflits et de défauts. Cela peut en effet engendrer des données imprécises, erronées, périmées, en contradiction ou incomplètes. L'incertitude est donc un problème majeur, mais comment le robot peut il acquérir et comprendre le contexte malgré cette incertitude ? Une fois le contexte perçu, le robot entre dans la phase de cognition pour raisonner et prendre une décision. Ce dernier doit comprendre et identifier si la situation est anormale, et si c'est le cas, intervenir pour résoudre cette situation. Ainsi, comment le robot peut détecter des situations anormales, à l'exécution et malgré l'incertitude, afin de prendre une décision ? Pour atteindre l'objectif choisi par sa décision, le robot va établir un plan d'actions via un planificateur. Un tel outil se base sur une connaissance préétablie. Cependant, chaque environnement étant différent et ouvert, il est irréaliste que cette connaissance soit exhaustive. En plus de cela, pendant que le robot suit son plan, des changements peuvent avoir lieu dans l'environnement. Ces deux faits engendrent des échecs d'actions, ce qui n'est pas acceptable. Comment le robot peut-il donc comprendre les causes d'échec pour ensuite les éviter proactivement ?

Dans cette thèse, nous avons proposé plusieurs contributions, s'appuyant sur le raisonnement et/ou l'apprentissage. Pour faire face au problème d'incertitude, nous avons proposé d'enrichir sémantiquement et de flouter des événements fournis par les capteurs. Cela permet à notre approche de gérer quatre dimensions d'incertitude, là où les méthodes classiques n'en aborde que deux ou trois. Pour la détection des situations anormales, et par la suite la décision, nous avons étudié une méthode hybride basée sur les Réseaux Logiques de Markov et une ontologie de contexte. Grâce à une sémantique de plus haut niveau dans l'inférence, nous arrivons à détecter précisément les situations anormales, et ce, même si la connaissance est partielle. Enfin, afin d'éviter les échecs d'actions dus à un manque de connaissance, nous avons proposé une méthode d'apprentissage par renforcement qui identifie les causes d'échecs et enrichit la connaissance du planificateur. Quant au dynamisme de l'environnement, alors que les approches classiques attendent l'échec pour recréer un plan, nous avons conçu un outil qui crée le plan incrémentalement et l'exécute à la volée. En combinant ces deux propositions, les échecs peuvent être proactivement évités. Au final, toutes ces contributions peuvent être regroupées pour former un framework global qui peut être spécifié et configuré pour de multiples robots et environnements intelligents.

Chacune de nos contributions a été implémentée, testée et validée au travers de simulations et/ou de tests à l'aide d'un petit robot humanoïde et d'une plateforme intelligente.

Contents

List of Figures

List of Tables

Acronyms

ADL Action Description Language.

ADL Activity of Daily Life.

AI Artificial Intelligence.

ANN Artificial Neural Networks.

ASD Autism Spectrum Disorder.

ASP Answer Set Prolog.

BN Bayesian Network.

BT Behaviour Tree.

CAREDAS Context and Activity Recognition Enabling Detection of Anomalous Situation.

CEP Complex Event Processing.

CFG Context Free Grammar.

CoDAMoS Context-Driven Adaptation of Mobile Services.

CONON Context Ontology.

CRF Conditional Random Field.

DHTN Dynamic Hierarchical Task Network:.

DST Dempster-Shafer Theory.

DT Decision Tree.

ETOILE Espaces et Technologies Ouverts pour l'Innovation des Laboratoires et des Entreprises.

FAIRIE For An Integration of Robots in Intelligent Environments.

FFCA Fuzzy Format Concept Analysis.

FSCEP Fuzzy Semantic Complex Event Processing.

FSM Finite State Machine.

HATP Human Aware Task Planner.

HATP Hierarchical Agent-based Task Planner.

HFSM Hierarchical Finite State Machine.

HMM Hidden Markov Model.

HRI Human Robot Interaction.

HTN Hierarchical Task Network.

ID Influence Diagram.

ILP Inductive Logic Programming.

INCOME INfrastructure de gestion de COntexte Multi-Échelle pour l'Internet des Objets.

IoT Internet of Things.

LEAF Learning, Evaluating and Avoiding Failures.

LIREC Living with Robots and Interactive Companions.

MEBN Multi-Entity Bayesian Network.

MLN Markov Logic Network.

OWL Web Ontology Language.

pDB Plan DataBase.

PDDL Planning Domain Description Language.

PEIS Physically Embedded Intelligent Systems.

POMDP Partially Observable Markov Decision Process.

RDF Resource Description Framework.

REST REpresentational State Transfer.

RFID Radio Frequency Identification.

ROS Robot Operating System.

SCFG Stochastic Context Free Grammar.

SHOP Simple Hierarchical Ordered Planner.

SHOP2 Simple Hierarchical Ordered Planner 2.

SLAM Simultaneous Localization And Mapping.

SSNO Semantic Sensor Network Ontology.

STRIPS Stanford Research Institute Problem Solver.

SVM Support Vector Machine.

UBS Utility Based System.

UCB Upper Confidence Bound.

UNR-PF Ubiquitous Network Robot Platform.

UPnP Universal Plug and Play.

VARSeR Visual Activity Recognition with SEmantic Reasoning.

Chapter 1

Introduction

1.1 Research Context

Who hasn't dreamed of being served by robots at home ? Or being helped ubiquitously by various devices in the environment ? Over the years, people have kept foreseeing what could be possible and how our life could be improved with robots and intelligent spaces. From Asimov's robots to Iron Man's JARVIS ubiquitous intelligence, through Disney's Wall-E interpretation, these technologies are now part of the collective imagination, for the good and the bad. For a long time they were purely fiction. However, with the current breakthroughs in robotics, Artificial Intelligence and Internet of Things (IoT), they are getting closer to reality. Nowadays, we observe the emergence of **service robots** in museums, train stations[1], and at home. Similarly, we can see our environments getting smarter thanks to novel sensors and actuators, particularly for homes, also known as **smart homes** in that case. These new products are also relatively immature and primitive. Huge progress are under way as there is a growing demand. In fact, although robots and smart homes can improve comfort and security for everybody, such technologies are also answering a major society problem: ageing[2]. With the population growing older, robots and smart environments offer a promising opportunity for cheap domestic health care[3] [4] [5]. In fact, smart homes and robots can provide user monitoring, perform daily tasks, ensure surveillance, and keep company against loneliness.

In this thesis, we want to study the management of data for both robots and smart environments in shared domestic context. To acquire and use data, robots and smart environments can rely on numerous method and techniques, each having their own features, strengths and weaknesses. But where are we today in these research fields ? What are the problems we are facing ? They both share the same purpose: enhancing users' life. They sometimes operate independently from one another, but wouldn't it be more relevant for them to work together, to cooperate? In the next subsections, we discuss these matters after defining and clarifying what are personal robot and smart environments, including smart homes.

[1] https://www.ter.sncf.com/pays-de-la-loire/gares/services-en-gare-et-a-bord/pepper

[2] http://www.un.org/en/sections/issues-depth/ageing/

[3] https://www.youtube.com/watch?v=ppPLDEi82lg

[4] https://www.youtube.com/watch?v=XuwP5iOB-gs

[5] http://medicalfuturist.com/healthcare-is-coming-home/

1.1.1 Personal Robots

Robots have been studied and developed for decades. Over the years, they have acquired numerous forms and purposes. Some of them are industrial arms in factory, others are floating assistant in the International Space Station, while others take the shape of a plush[6]. To classify these robots, Kim [91] pointed out three generations of robots matching the eras of modern computing:

1. Industrial Robots, that aim to execute repetitive tasks in factories.
2. Personal Robots, that help persons in their everyday life.
3. Ubiquitous Robots, that can perform services anywhere, any time.

First generation robots, that is to say industrial robots, started emerging during the fifties and are nowadays mature and widely used in factories. They aim to perform precise manufacturing tasks that are tough and tiring for humans. These robots are the achievement of years of innovation in mechatronics, but they do lack intelligence. Current research is pushing toward smarter industrial robots that have the ability to collaborate with humans, also known as cobots or collaborative robots [136, 52]. Although they are sometimes used with specific external sensors [42, 13, 2], industrial robots rarely share their environment with other devices. The industry 4.0 pushes toward more smart environments around those robots, but remains limited.

At the opposite of the spectrum, we can find ubiquitous robots [92]. These third generation robots are able to navigate the ubiquitous space to provide services anytime, anywhere, on any available devices. In fact, such robots are part of a large smart space and can go through networks to use devices according to their needs, such actuators in a smart home or robotic "bodies"[7][8]. Some researchers have already started exploring the third generation of robotics [91, 92], including projects such as LIREC[9] [61] or UNR-PF [85]. However, this is a relatively novel field of research with major bottlenecks to pass [34].

Today, research and industrial effort is mostly being put in the second generation of robots, namely personal robots. As implied, such robots aim to provide personal help and service in daily living, in particular at home. The research community is noticeably active around personal robots, as illustrated by the Robocup @Home competition [186]. In fact, this generation is pushed forward by societal demands, for domestic services, but also and mainly for health care. As a matter of fact, personal robots are explored to help disabled people [64, 178] and persons with Autism Spectrum Disorder (ASD) [127]. On top of that, personal robots were extensively studied through major projects such as Robot-Era [22] or CompanionAble [67]. In this thesis, we are considering **personal robots** in a domestic context. But what is a personal robot?

As implied, a **personal robot** is, first of all, a robot: it is a device equipped with sensors, actuators and computational units that allow it to perceive, navigate, act and interact with humans. Personal robots are conceived to help one or multiple persons by performing tasks according to the will and/or

[6] http://www.phoque-paro.fr/
[7] https://vimeo.com/21156543
[8] https://www.youtube.com/watch?v=uKL0Jbn_6nc
[9] http://lirec.eu/

need of users. To this end, the robots need to be provided with a high level of intelligence, as well as, the ability to interact with humans, referred as Human Robot Interaction (HRI). This kind of robots can also be referred as **service robots** or, in some cases and particularly for vacuum robots, **domestic robots**. In the rest of this document, we will refer to them by using equally **Personal Robots** or **Service Robots**.

Personal robots, thanks to their mobility, sensors and actuators, are able to perform complex tasks anywhere in the environment. They can, for instance, patrol in a house for surveillance or grab objects to deliver to the user. However, some tasks, yet straightforward for humans such as opening doors, can be challenging for robots.[10]. Furthermore, another issue is the range of their sensing abilities, in fact, they can only be aware of their immediate surroundings and may have problems monitoring distant elements. These are possibly strong limits for personal robots. Yet, they could be solved by a smart environments providing more information and possibilities to a personal robots.

1.1.2 Smart Environments

Along with the emergence of Artificial Intelligence (AI), smart environments, in particular smart homes, are becoming more and more popular [169, 105]. A smart home consists of a housing equipped with smart devices: electronic devices that can sense, communicate with other smart devices, and act with some intelligence, for instance, an actuator may perform a task under particular conditions. Smart devices differ from classical devices by their intelligence and communication capabilities, for example, a smart fridge[11] is not only able to store food, but is also connected to the internet and can interact with a smart phone. More generally, a smart environment is a place, such as a shop, a museum [4] or other, that is equipped with smart devices. Some companies are proposing all-in smart home solutions, such as Orange[12]. Nowadays, we observe the emergence of the Internet of Things (IoT), where objects are connected to the Internet. Pushed by this rise, there is an increasing number of various smart devices on the market, ranging from toothbrushes[13] to cameras [14]. Yet, these newly available sensors can be used locally for the intelligence of the room. Be aware that this work is not more related to the IoT than the usage of similar devices.

The research community about smart devices network is highly active. In fact, **pervasive or ubiquitous computing** is a trendy topic [190, 189, 145, 12], yet much larger than smart homes: pervasive or ubiquitous environments refer to environments that can perform services anywhere, anytime, which is out of the scope of this work. The literature is also active on more specific environments enhanced with devices, that are referred to as **smart environment** or **ambient intelligence** when referring to the behaviour mechanism [12]. As a smart home is a housing enhanced with devices, a smart home is a smart environment. In this document, we will equally use the term **ambient intelligence**, **smart environment**, as well as, more specially, **smart home**. Note that, in this work, although we focus on domestic application, we consider all kinds of smart environment, in particular in our review of the literature.

[10] `https://www.youtube.com/watch?v=UUOo8N9_iH0`

[11] `https://www.samsung.com/us/explore/family-hub-refrigerator/overview/`

[12] `https://homelive.orange.fr/`

[13] `https://www.kolibree.com`

[14] `https://www.netatmo.com/en-GB/product/security/`

At home, ambient intelligence enables automatic control, for lighting and heating for example. It is also an efficient automatic security system, for instance, Netatmo Welcome is able to identify users and triggers an alert if the person is unknown. Smart environments also offer promising possibilities for health care [105]. In fact, they are a proper solution to monitor patients: medical staff can be informed of the activity of the user or health information, and be alerted if necessary, if the user falls for instance. Smart environments can also perform specific tasks, turning on the light in the most obvious example, but we can also think about automatic doors or smart TV. However, the sensor network may not cover the whole home, typically, the user may fall in a blind spot. Furthermore only specific tasks can be performed. On top of that, it may be uncomfortable for a person to interact with a smart environment. In fact, it has no real face or body to represent it and this can be disturbing for the user. These issues could be solved by interacting with a personal robot, thanks to its mobility and ability to naturally interact with humans.

1.1.3 An Illustrative Example of a Personal Robot Interacting With a Smart Home

As we pointed out, both smart environments and personal robots are emerging and provide interesting possibilities of domestic services, including health care. Each of them has its own strengths and weaknesses, they however fulfil the same role: improve user's quality of life. Consequently, combining both seems to be a promising approach. In fact, by interacting with each other, personal robots and ambient intelligence can overcome the weaknesses of each other and enable more possibilities, providing an overall improved quality of services. However, such an interaction can also induce new challenges. Let us have an illustrative example that depicts the interest and challenges of using personal robots and smart environments.

Arthur and Nono

Arthur is an elderly living alone in its smart home. Actually, he is not exactly alone as he possesses his own service robot: Nono. Nono helps Arthur's daily routine by performing multiple and various tasks, from information to object manipulation. The robot can rely on the smart home and its several devices to achieve its goals. In fact, the smart environment contains various sensors; it is composed of cameras (2D/3D), movement sensors, beacons, RFID tags and sensors, opening sensors (on doors and windows), Arthur's smart phone, connected devices (stove, TV, fridge, etc.), thermometers, gas detectors, air quality sensors, switches, and microphones. It also possesses various actuators: lights, smart devices (TV, fridge, heaters, etc.), motorized doors, alarms (i.e. speakers), motorized windows and shutters, and user's smart phone. Nono, the personal robot, is equipped with various sensors, including 2D and 3D cameras, sonar, microphones, inertial units, contact sensors, and internal monitoring sensors (battery, temperature, etc.). It has arms that allow to grab and carry small objects or press on buttons. Nono embeds a basic intelligence and is able to speak, listen, and identify user's activity using its cameras.

A Daily Adventure

It's almost noon. Arthur, as usual, is preparing his lunch. In the meantime, Nono is standing by in the living room. Arthur's cooking implies heating a meal for a long time, thus, until it ends, he decides to lay down on the living room sofa. However, after a few minutes, Arthur fell asleep. Nono was looking at him, thanks to its cameras, it guesses that its master was sleeping. This is confirmed by the living room microphone that detects low noise level. Thanks to its connected stove and thermometer, the ambient intelligence detects the stove is still on, and informs Nono about it. This could be a problem, as the meal may be overcooked or a more hazardous event could occur, such as a fire. As Arthur is sleeping, Nono decides to directly go to the kitchen and to shut down the stove. The robot computes a plan of actions to achieve it: it has to (1) go to the kitchen, (2) approach the stove, (3) shut it down and (4) activate the kitchen vent through the smart environment. It proceeds to execute the plan and reaches the stove. As the stove has no connectivity, the robot then presses the easy manipulable button to turn the stove off Lastly, it communicates with the smart environment to turn the vent on. Nono has successfully prevented any hazard and cleared the air in the kitchen. After this task, the robot decides to go back to the living room and informs Arthur of its action as soon as he is awake. By the time Nono reaches the room, Arthur has woken up and moved to the bedroom, while still having forgotten his meal. When it arrives in front of the sofa, Nono couldn't find Arthur, and apparently, no movement was detected in this room. However, the ambient intelligence has detected movement in the corridor and the bedroom. Nono infers that Arthur is likely to be in the bedroom and proceeds to go there. On the way, the bedroom's door appears to be close, but Nono stops its current action to ask the motorized door to open, then the robot enters as it was expected to do. Once there, it successfully finds Arthur and explains him what it did. Arthur is happy with what Nono did and immediately proceeds to the kitchen to save his lunch and finally eats something...

Discussion

This scenario highlights the advantages of interactions between a personal robot and a smart environment. First, it provides a better quality of the user's activity recognition. By using both the robot sensors and the smart home sensors, it can reinforce the knowledge. There is a "double check" that enables a better understanding of the context, in this case the activity recognition. Moreover, without the smart devices, Nono would not have been able to locate Arthur when he left the living room, nor Nono will be able to move around in the house. Combining smart home and robot provides a better coverage thus allowing a better quality of service; Nono knows where it needs to go and it can open more easily a door by activating the door actuators. In brief, Nono is more effective and its actions more accurate. **From these ideas, in this thesis, we want to explore the possibilities of personal robots in smart homes.**

Unfortunately, we are currently still far from the ideal scenario described above. The interaction between robots and ambient intelligence, although beneficial, involves addressing numerous obstacles, such as the dynamism of the environment, the imperfection of sensors and heterogeneity of data. Furthermore, decision making and task planning, that are essential for Nono, are also demanding and complex process that additionally requires to take into account both robot and smart

environment. In this thesis, we aim to tackle these challenges. Let us now review them.

1.2 Challenges and Objectives

As illustrated in the scenario, robots and smart environment have the possibility to provide a high quality service to user. However there are many and various challenges to overcome before achieving such performance. Before addressing the challenges, we use a simple and common abstracted architecture based on three layers: **perception**, **cognition**, **action**. This architecture allows to classify and position challenges and works in the global process of the data. Note that this general pattern carries no novelty. In fact, many researchers have proposed architectures [3, 104, 33, 19, 150], yet the general pattern perception, cognition, action can be observed in those works. Furthermore, more layers are sometimes considered, such as "empathy"[15], but are not necessary for us. Let us see the three steps we consider:

- **Perception**: corresponds to the context data (or contextual data) acquisition. First, raw data provided by the robot and/or environment sensors are collected. By combining these sensor data and by correlating them towards other semantic data sources, higher context data are inferred.
- **Cognition**: corresponds to the analysis of a situation at broad. The cognition consist in abstracting context data into information, then knowledge through the understanding of situations. It enables the robot to understand what is happening in order to take actions. This includes the user activity recognition (i.e. sleeping, cooking). An activity itself can be composed of sub-actions such as walking, grabbing an object. By recognizing a predefined situation, that is a state of context, one is able to make a decision to react. This relates to the intelligence of the robot and/or the smart home.
- **Action**: once the robot and the ambient intelligence know what to do, they can act on the environment. To this matter, task planning is commonly used. It is a method that generates a sequence of tasks according to the context. An action can be an order given by the user, the activation of some actuators, or the mobility of the robot. Again an action might be complex and decomposed into many sub-actions, such as going to the kitchen, grabbing the bottle, etc.

In order to provide an efficient and satisfactory service, personal robots and smart robots need to tackle these three layers. We identified multiple challenges, belonging to one or two layers, that are to be tackled:

1.2.1 Perception Problems: Heterogeneity and Variability of Sources of Context Data

Acquiring the knowledge of the context from sensors and data sources is an essential step of **perception**. Yet, for a robot in a smart environment, it is subjected to the challenge of heterogeneity of context data. In fact, the environment is filled with various devices, on top of the robot. In the

[15] `http://www.bbc.com/news/video_and_audio/must_see/40306617/five-robots-that-are-changing-everything`

scenario, the Arthur's home is filled with thermometers, microphones, motion sensors, smart stove and more, Nono is also equipped with cameras, microphones and sonar, each working differently and providing various context data. Thus, Nono needs to acquire multiple types of data from various type of sources. Data sources can provide data in different ways. Context data can be transmitted in stream, as events, pushed or pulled. Although it is possible to transform one type of data transmission to another, in the end, the robot shall be able to use any of them. Note that in this thesis, we suppose context data provided through middleware, how the devices communicate is out of the scope of this work. We suppose all the context data to be provided as events, meaning they are pushed and associated to timestamps. We actually focus on the heterogeneity of data. In fact, the robot needs to consider different types of data similarly, for instance the user location and the temperature have different format and meaning, yet must be taken into account similarly. This implies using a **rich context model that supports any type of data**. Ontologies are known for offering such a feature: their triple based design permits to associate entities with properties of numerous types.

1.2.2 Perception and Cognition Problems: Dealing with Uncertainty

Sensors are not perfect: they can be mistaken and provide uncertain data. This can lead to an erroneous knowledge of the context and to an inadequate response from the robot. From example, if Nono receives context data indicating Arthur is in the living room while he left to the bedroom, Nono will try to look for him in the wrong room. This could at best delay the robot or, at worse, causes it to fail its task. Thus, uncertainty is a problem of **perception** and **cognition**. We consider five dimensions of uncertainty [190]:

- **Freshness** represents the temporal relevance of a context data. In fact a context data is valid for a given duration and may become obsolete after this delay. For example, if Arthur moves from the living room to the bedroom, the context data stating he is in the living room is outdated, thus wrong.
- **Accuracy** represents the "truth" of a context data. Indeed, due to hazard in the environment or miscalculation, sensors have sometimes mistakes and may provide information that are actually wrong. For instance, a low quality thermometer may provide an aberrant temperature of 150°C.
- **Precision** models the exactitude of a context data: an imprecise data is true, yet inexact. Motion sensors are a perfect example for imprecise data; such a device only delivers a signal meaning a movement occurred, but it is unable to tell who was making the movement and where he/she/it was in the environment.
- **Contradiction** is the problem of having two context data providing contradictory information. One (or the two) sources may be wrong or the two may be partially true, e.g. due to imprecision. For example, Arthur may be perceived in two rooms at the same time by different sensors, this can be due to one being inaccurate, or Arthur being actually between the two rooms.
- **Completeness** represents the availability of context data. By not being available, due to a breakdown for example, the absence of a data in the model could lead the cognition to be

inaccurate. In Arthur's home, a motion sensor may be broken down, thus not providing location information.

In order to have a proper context knowledge, **it is important to cope with these uncertainty dimensions**. This implies a particular acquisition process as well as a suitable context model. Otherwise, the decision making and task planning could be negatively affected, leading to inadequate decisions and task failures. In the literature [190], there exist various solutions tackling uncertainty dimensions. Yet, they only tackle a few dimensions at a time. Tackling all these dimensions at a time is a major limit in the state-of-the-art.

1.2.3 Cognition Problems: Context and Situation Awareness

Once the context is acquired, the robot can decide to intervene by itself. It needs to be intelligent and to correctly understand the surroundings. For example, when Arthur forgets he was cooking and goes to sleep, there is a risk of the kitchen getting smoked, or, at worst, a fire can occur. In consequence, Nono decides by itself to intervene to shut down the oven. Detecting such **anomalous situations** is a major challenge of the **cognition** layer. In fact, it requires a rich knowledge including complex context data, such as the activity recognition, that are not obtained directly from sensors, but are inferred. In consequence, **the data model must support these kinds of data and be able to generate them**. Ontologies are famous for their compatibility with reasoning engines and can provide a proper solution. However, from this knowledge, the situation awareness should be enabled. In other words, the robot should understand the situation and accordingly take a decision. This requires strong reasoning abilities based on learning or rules and using all of the context knowledge. In fact, most works actually only consider the activity of the user to detect the anomaly [75, 147, 79] and do not consider possible anomalous situations from other sources of data. As mentioned earlier, the uncertainty of data has an influence on the process and shall be considered. **The detection of situations when the robot should intervene over a complex context knowledge is a challenge to be addressed**. Note that we consider only one robot in the environment and the decision making is considered to be centrally performed by this robot.

1.2.4 Action Problems: Risk of Failure

Once the robot has decided what to do, or received order from the user, it proceeds to the **action** layer and determines a course of actions, or tasks, to follow in order to execute the decision. To do so, it can rely on a task planner. A task planner is a tool that generates a sequence of tasks to reach a given goal according to the current context. Hence, it finds a solution adapted to the current context. Typically, when Nono decides to shut down the oven, it generates a plan of actions that consists of first going to the kitchen, then approaching the oven, pressing the oven switch, leaving the kitchen, going to the bedroom and finally warning Arthur. There exist multiple tools to plan; they mainly rely on chaining based on preconditions and postconditions, or task decomposition into subtasks. Once the plan is generated, the robot executes it task per task. However, the robot may fail some tasks, thus impacting the whole plan. In fact, in domestic application, it can encounter various problems causing the failure of the task including:

- **Breakdown**: the robot has a breakdown or an internal error causing it to be unable to perform the task. For example, if Nono has its arm engines overheating, it is unable to grab objects.

- **Robot limits**: the robot is given a task, but doesn't have the capabilities to perform it. Typically, Nono can be asked to grab an object on the table while its arms are to short to reach it.

- **Dynamism of the environment**: the environment, thus the context, varies over time. Executing a task and, by extension, a plan is time consuming, for example, for Nono, going from a room to another is a matter of minutes. In the meantime, many changes can occur, leading to the plan to be outdated, as it was generated according to a now obsolete context, leading to failure. For example, when Nono has to warn Arthur who is in the living room, by the time it reaches this room, Arthur may have moved to the bedroom, causing Nono to fail to reach Arthur.

- **Unexpected conditions**: when planning, the task are selected according to conditions. These conditions define if the task can be performed. However, it may happen that these conditions are incomplete according to the environment, thus causing a task to be picked even if it will fail. For example, with Arthur, Nono has no problem to vocally warn the user, however, if we consider another user, Jeanne, that is deaf, she won't be able to hear it, making Nono fails as the task of vocally warning is not applicable with a deaf user. The implied problem is that an expert can hardly provide conditions exhaustively taking into account possible failure causes, as it varies according to the environment, user and robot.

Failures are a common problem for personal robots. A commonly used solution is to regenerate a new plan when failure is encountered [185, 116, 20], yet this is costly in execution time, energy, and may give a bad impression to the user. **Thus, failures are to be proactively avoided, which is an open challenge**.

In our case, the robot is not alone, it also interacts with a smart environment, for acquiring the context, but also for acting. Most planners don't take into consideration the sources of data. Yet, intensively smart devices can exhaust their energy, for example by constantly querying Arthur's phone for movement context data, Nono may empty its battery. However, not all context data are useful to generate a plan. On top of that, the plan may use smart devices to act, however, they may be already in use or busy. For example, if a heater is being adjusted by Arthur, it should not be used in the plan. The smart environment is indeed a strong source context data for the robot, but it is also a source of constraints when using it. **In the end, taking into consideration the constraint of the smart environment in the task planner is important, yet novel**.

1.2.5 Transversal: Open Environment

A constraint is the open aspect of the smart environment. In fact, smart homes and robots are various and equipped differently. Arthur's home is not the same as everyone else's. Each environment has its own devices and possibilities. Some may have a rich coverage of sensors, others may have limited sensing opportunities, but more actuators. Hence, **our propositions must be compatible and**

independent of any environment. For example, if Nono was sent to another smart home, with different sensors and actuators, it should be able to operate after resetting with the help of an expert. Thus, Nono must not be dependent on specific devices, for example, a solution using only audio stream for surveillance would be incompatible with environments and robots with no microphones. Furthermore, as the environment is open, new devices may be added at runtime. Typically, Arthur may decide to buy a new device or, oppositely, remove one from the environment. **It is important for our solution and our model to be able to evolve and adapt as the smart environment changes**. For example, if Arthur adds new beacons for localization, the knowledge of Nono needs to be easily adapted in accordance. Handling the open aspect of the smart environment is a constraint that impacts all other challenges during perception, cognition or action.

1.3 Contributions

For robots within smart environments, it is essential to have a great management and understanding of data in order to ensure the quality of service to users. However, with current research advancement, numerous challenges are remaining unsolved. In this computer science thesis, we proposed multiple contributions to cope with those challenges and to provide solutions for data management for robots in smart environments. Through these works, we explored the usage of various techniques and combinations, including learning and reasoning techniques. We can divide them in four steps that follow the perception, cognition and action process:

1. First come context acquisition and understanding contributions, that aim to solve perception challenges, including uncertainty and heterogeneity of context data, to enable a proper context knowledge from the sensors.
2. Secondly, cognition propositions are applied on the acquired knowledge in order to make a decision. This requires the understanding of the situation while supporting uncertainty.
3. Thirdly, once the decision is taken, the robot can act by using our planning contribution, that aims to tackle the action problems through a novel planning paradigm.
4. Finally, as limitations of planning knowledge can cause failures, experience can be acquired from the robot actions to perform cognition for the robot to learn and enhance the knowledge of its planner.

As a whole, our contributions compose a larger system: **the FAIRIE framework**. Let us have an overview of our contributions.

1.3.1 Perception: From Sensors to Knowledge

In fact, perceiving through sensors is key to achieve the knowledge of the context, that is the root for the intelligence of robot and smart environment. Acquiring the context in a domestic environment is not a simple task as data sources are imperfect and various. While most most sensors can be assimilated as event based sources, others require particular processes, such as robot cameras. Hence, we proposed two contributions to acquire the context under the constraints of smart environment.

Firstly, we tackled context acquisition based on events. Our contribution, FSCEP gathers, batches, semantizes and fuzzifies events provided from various sensors, from the smart environment or the robot. By applying fuzzification on batch of events, it generates a single high level event carrying a fuzzy context data. This is enabled by relying on ontological background knowledge as well as Complex Event Processing (CEP) and fuzzy logic. Our contribution is consequently capable to handle various types of events, thus context data, and supports four uncertainty dimensions, namely freshness, accuracy, imprecision and contradiction. Furthermore, FSCEP is a specification approach and can easily be extended by adding new rules.

Secondly, we addressed the case of vision based acquisition, in particular for activity recognition. In fact, user activities are essential context data. One option to acquire it, is for the robot to rely on its camera to recognize and classify the gesture of the user. The idea is to analyse a video stream of the user to infer his/her activity [46]. However, although this learning-based approach offers decent results on datasets, in more general cases, it has difficulty to recognize and usually provides an inaccurate and imprecise activity recognition. In order to overcome these limits, we propose a combination between vision based activity recognition and ontological context knowledge obtained from the smart environment. With this knowledge, our contribution, VARSeR, refines the output of the vision based approach into more precise and accurate activity recognition.

1.3.2 Cognition: From Knowledge to Decision

Cognition allows to understand the context and make a decision from it. In our work, we are particularly focused on anomalous situations, that is to say situations that require robot intervention. Understanding the situation requires a strong analytical process and a support of data uncertainty in order to provide an accurate decision. To deal with these **cognition** challenges, we propose CAREDAS, that uses both Markov Logic Network and ontologies. CAREDAS uses a semantic context knowledge and applies a complex inference process based on MLN. Unlike most approaches, it is able to detect anomalous situations over the whole context, and not only on user activities. Our solution is able to take into consideration the uncertainty of context data by adjusting weights accordingly. On top of that, the process does not require an exhaustive observation of the situations and consequently supports completeness. This enables CAREDAS to accurately detect anomalous situations. From this identification, the robot can decide which goal to achieve in order to "solve" the detected situation.

1.3.3 Action: From Decision to Actions

In order to act, the robot uses a task planner. This allows it to determine a sequence of actions to follow in order to reach a goal according to the current context. However, during the execution of the plan, the context may change, causing failures. To cope with this **action** layer problem, we proposed a novel planner DHTN that combines execution and planning. DHTN enhances a commonly used planner, HTN [170, 43], but uses a different planning paradigm. In fact, instead of generating then executing the plan, DHTN incrementally and alternatively generates and executes the plan. By doing so, **DHTN can take into account the latest context changes when planning**. This implied not only creating the plan, but also monitoring it and correcting it if necessary. Furthermore, DHTN is capable

of computing a subset of context data needed for planning. From there, only the necessary context data are used, preventing waste usage of sensors. On top of that, it uses statuses to monitor and control the availability of actions, in particular through smart devices. This makes DHTN thrifty and efficient in a smart environment.

1.3.4 Cognition/Action: From Actions to Experience

Even if DHTN is able to cope with the dynamism of the environment, the planning knowledge can hardly be exhaustive and failures may be caused by unexpected conditions. LEAF is our **cognition** and **action** contribution that aims to enrich the planning knowledge from experience, thus proactively preventing further failures. LEAF aims to identify failure causes from the experience of previous failures. To do so, an history is created from previous encountered situations using ontologies. From there, we used a reinforcement learning approach that includes the user in the loop, ensuring the quality of the learning. We modelled this process as a multi-armed bandit problems. Consequently, a state-of-the-art multi-armed bandit approach, R-UCB [28], was used and improved. In fact, in order to refine the learning, we used causal induction and causal graph to determine direct and indirect failure causes. From this learning, LEAF provides a knowledge of the causes that can then be used to avoid further failures: if a task failure cause is observed, then it shall not be used in the task plan as it is likely to fail. By doing so, LEAF proactively prevent task failures due to unexpected conditions, but also the ones, in particular cases, due to robot limitation or breakdown.

1.3.5 FAIRIE

All in all, these contributions form a bigger framework called FAIRIE. FAIRIE is a set of tools, including but not limited to our contributions, that allows to deal with perception, cognition and action for a personal robot in smart environment. FAIRIE needs to be parametrized and defined according to the environment, thus being a framework, but can be adapted in various environments. In fact, according to the needs, modules, that are our contributions, can be used or not. For example, FAIRIE may be configured for an environment that only includes a robot, or can be set to use all its capabilities in a rich smart home with a strong personal robot. **This enables FAIRIE and our contributions to be compatible with a large variety of environments**.

1.4 Thesis Organization

This manuscript describes the work conducted during my thesis and is divided into four parts. The first part reviews the state-of-the art. After an introductory chapter, we review the existing combination between **robots and smart environment** and identify the current and open fields of problem. From there, two chapters review respectively and specifically **context awareness techniques** and **task planning**. We then address our contributions in two parts respectively corresponding to context awareness and task planning in the resulting framework: FAIRIE. In the second part, we review our perception contributions that handle data **from sensors to knowledge** and that enable the robot to go **from knowledge to decision**. In the third part, we present how our novel planning approach

can turn the **decision to actions** and how the **experience from actions** can be used to enrich the planning knowledge. In a conclusive fourth part, we provide more information about **FAIRIE and its experiments**, before **concluding** this document by discussing the lessons we learnt and providing various perspectives.

Part I

State-of-the-Art

Chapter 2

Introduction

Robotics and ambient intelligence can both rely on active research communities. Each of these fields carry a tremendous amount of challenges and topics, going from hardware conception to self configuration through computer vision. Combining robots and smart environments seems to be a promising approach for various needs, including domestic healthcare. For this reason, many researchers and projects explored this idea. Although such an approach has specific challenges, they are also numerous, some being more mature than the others.

In this part, we aim to review the literature, to evaluate the current state of the research around robot in smart environment, and to identify the precise limitations to tackle in order to achieve our objectives. As the topics are miscellaneous, we first review the existing works of robots collaborating in smart environments. From this analysis, we position ourselves according to other works. In fact, robots can collaborate with devices in various contexts, yet, in our work, we focus on service and domestic context. Then, we identify fields of problems encountered by these restrained works and evaluate their advancements. This allows us to identify two fields that are facing limitations toward our objectives: context awareness and task planning. Context awareness incorporates methods used to acquire, model and use the context, which is essential for domestic services. Task planning aims to generate a sequence of tasks for the robot to follow in order to fulfil an order or a decision according to the context. In fact, task planners rely on the knowledge acquired from context aware techniques, making the two closely related while having different research communities.

In the next chapter, we review the challenges encountered by personal robots in smart environment and discuss how researchers, through projects, addressed them. In the following chapter, We then shift our focus on general context awareness techniques We point out strengths and limits of those techniques when applied in our case. Lastly, in the third chapter, we review existing task planning solutions and discuss them against our requirements. Again, we extract weaknesses in the current planning technologies that need to be solved. This part ends with a conclusion summarizing our analysis of the literature.

Chapter 3

Personal Robots In Smart Homes

3.1 Introduction

Both personal robots and smart homes are common subjects in research. Although, each of them has its own focuses and problems, they do share a same purpose: improve user life. Consequently, the research community has explored the possibility of using robots and smart environments together. In this chapter, we aim to discuss the fields of problems encountered by personal robots in smart homes from a global scope. As we will see, among the described and addressed problems in the literature, our challenges are not tackled. In order to understand the challenge they may face, it is essential to understand personal robots capabilities. Consequently, we firstly describe today's personal robots. From this description, it can be observed that such robots can encounter difficulties for acting and sensing. Smart homes can provide a solution to these problems, although they may also induce new ones. The interaction between personal robots and smart environments was explored through multiple major research projects. Those projects aimed, and actually achieved to some extent, to deploy robots in domestic smart spaces. In consequence, they tackle a complete spectrum of problems to which personal robots in smart environments are subjected to. Accordingly, we will briefly review and pinpoint these classes of problems. From this, we will underline which challenges are yet to be addressed in two particular fields.

3.2 Personal Robots

Robots are various in forms and purposes. Some are specialized in executing industrial tasks while others can embody multiple ubiquitous devices. In this work, we consider and focus on **personal robots**. As implied by their name, personal robots aim to provide service to public in everyday life. These robots actually operate alongside users to help them with usual and common tasks, hence, most of them are also service robots. Unlike industrial robots, personal robots are intelligent, autonomous and often mobile. Such features have been enabled thanks to a large research effort over the past twenty years in numerous domains including sensing, navigation and AI.

[1] `https://www.ald.softbankrobotics.com/en/press/gallery/pepper`

[2] `https://zenbo.asus.com/product/gallery/`

(a) Softbank Pepper[1] (b) CompanionAble Hector (c) Asus Zenbo[2]

Figure 3.1: Examples of personal robots

After years of innovations and contributions, the first commercial personal robots are just being released, such as Blue Frog Buddy[3], Asus Zenbo[4], or IJini[5]. Furthermore, multiple robots were used for research in the past few years, such as CompanionAble Hector or Aldebaran, now Softbank Robotics, Nao. These robots can take various forms, as seen in Figure 3.1. Most of them rely on wheels, allowing them to easily navigate in smooth environments. Yet, some robots, such as Nao, use legs. They also usually feature a humanoid face, which is important for human robot interaction[6]: it allows user to naturally look and understand the robot. They may also be able to speak [82]. Recently, many personal robots are provided along with tablets, allowing easy interaction with a user and enabling the usage of visual messages. These robots are equipped with various sensors to perceive the environment. Cameras are usually the main sensing tools for these robots and the most recent ones even carry 3D cameras allowing them to use a 3D representation of the environment and recognize shapes. 2D cameras are also immensely useful, with the progress in computer vision, they can be used to identify the activity of the user [46], identify persons or even understand emotions [76]. Microphones are also widely used in personal robots, making them able to listen to users as well as other noises. Other personal robot sensors include sonar, range sensors, air quality sensors and others. Some of them are equipped with arms and hands allowing them to grab objects, but this technology is expensive, and many robots do not have any arms. Even if armless, these robots can already perform a companion, surveillance or assistant role. Other robots, as Pepper or Nao, are used for customer interaction in retails or in museums. For healthcare applications, robots are equipped with strong actuators and can perform possibly complex tasks. Finally, to enable sensing, control and interaction, these robots usually embed a computational unit able to perform operations independently from a remote computer. They are indeed able to apply vision algorithms, store data, compute trajectories, etc. In brief, personal robots vary in roles, capabilities, sizes and prices. They

[3] http://www.bluefrogrobotics.com/en/buddy/
[4] https://zenbo.asus.com/
[5] http://www.ipl.global/
[6] https://spectrum.ieee.org/automaton/robotics/humanoids/what-people-see-in-157-robot-faces

are mobile, intelligent and can perform numerous tasks. We consider this class of robots.

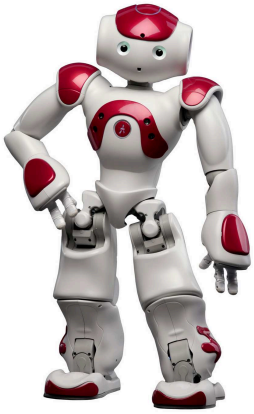


Figure 3.2: Nao robot[7]

In our work, we used the Nao robot [63], shown in Figure 3.2. Nao is a small humanoid robot popular in education and research. It is widely used in various domain including interaction with patient with ASD [165], locomotion [122] or computer vision [46]. Nao is a bipedal robot, which enables large possibilities of movement, but limits its ability to navigate on longer distance and time. With its arms, it can grab light and small objects. It can rely on two 2D cameras, as well as sensors including inertial unit, sonar, and tactile sensors. One the main strength of Nao is its API, Naoqi. In fact, Nao has the ability to talk, understand speech, identify persons and walk. Consequently, working with Nao is simple. This makes it a perfect experiment platform for us.

Although personal robots have been the subject of major effort, they are still facing several challenges, in particular in domestic context [186]. In fact, perceiving, thinking and acting are demanding and problems including localization, context acquisition or Human Robot Interaction (HRI) are currently hot research topics. Indeed, although they are equipped with various sensors and actuators, personal robots need to be able to use them correctly in order to be able to perform their function. Alongside personal robots, smart homes are also on the rise. As mentioned earlier, by interacting with each other, personal robots and smart environments could overcome some of their problems. Consequently, research has been exploring such a collaboration, in particular through major research projects such as Robot-Era [22] or CompanionAble [67]. These projects cover challenges encountered by personal robot and use smart environment In the next section, we will review these projects from a larger scope in order to identify the problems they encountered and tackled, but also their limits when facing the challenges pinpointed in our introduction.

3.3 Projects

Integrating robots in smart environment was addressed through multiple large projects. They tackled the aforementioned challenges and proposed solutions to various extents to solve them. These works are among the closest to our work as they explicitly used robots in smart environments, even if the problematic they address can be broader. Let us review these projects.

3.3.1 PEIS

The PEIS project [155, 31, 156] is undoubtedly a pioneer project for the integration of robots in smart homes. Similarly to us, yet more precursory, this project comes from the observation that homes are going to be equipped with various devices that "form an ecology of communicating and cooperating Physically Embedded Intelligent Systems (PEIS)". In such environments, each devices functionalities are improved by interacting with other devices. While most works tend toward complex and isolated

[7] https://www.ald.softbankrobotics.com/en/press/gallery/nao

companion robots, Saffiotti et al. [155, 156] opt for robots that are part of the environment. In fact, as depicted in Figure 3.3, PEIS is positioned at the balanced merge of robotics, AI and ubiquitous computing. Thus, robots are considered as any other devices in the smart environment.

The architecture and communication between devices are ones of the main focus of PEIS. PEIS relies on a distributed oriented middleware [32], where each component, devices and robots, perform tasks for the whole system. The configuration of the ecology is a big challenge for such a system. Thus, a centralized approach based on a hierarchical automated planner was studied [108], as well as a distributed approach that relies on recursively extend a provided initial configuration [65]. The project also tackles the issue of communication between devices, in particular the problem of anchoring [101]; in fact, as sensors are various and **heterogeneous**, determining data referring to the same object (i.e. anchoring) is a problem. As PEIS is aimed to be deployed in home, the HRI dimension of the project was also addressed [68].

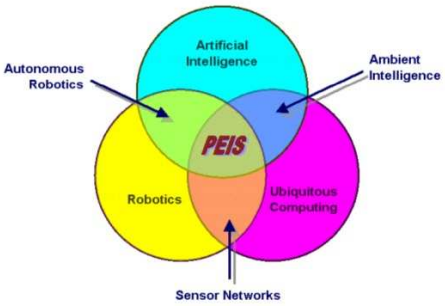


Figure 3.3: Positioning of PEIS, from [155]

Lastly, **task planners** were also studied within PEIS, even if it did not constitute a major axis of the project. For instance, PTLplan was used[87]; it is a probabilistic conditional planner that supports uncertainty. Moreover, the distributed system can monitor task executions of robots and/or devices and detects whenever a failure situation occurs.

3.3.2 Robot-Era

More recently, Robot-Era project[8] was conducted. Robot-Era is a European research project that aims to implement and to analyse advanced robotic system for elder care. This project is one of the most advanced application of service robot with rich and real-life experiments: it leads to tens of publications and is a practical illustration of robots integration in smart environments. Robot-Era relies on the contributions of the PEIS project as for the integration of robots in smart spaces, however, the problematic of the Robot-Era project goes far beyond. In fact, the contributions go from purely **technical integrations and configuration** [73, 40] to **legal, ethic and HRI aspects** of robots [21, 60]. Among the noticeable contributions, we can point out their proposition toward **robotic localization**. Actually, as already mentioned earlier, the usage of a RDIF floor to locate the robot was studied [89]. Another core contribution is Robot-Era **task planner**. Multiple dimensions were considered and resulting planner are able to cope with numerous constraints, including planning for multiple actors [41].

Figure 3.4: Robot-Era project logo

3.3.3 CompanionAble

[8] http://www.robot-era.eu

CompanionAble[9] is another important European project that aims to help people subjected to dementia and depression thanks robots and ambient intelligence [67]. Integration of robots in smart environments is an important matter of this work [15]. Yet, unlike PEIS approach, the CompanionAble robot has a lot of capabilities and can operate on its own [66]. Similarly to Robot-Era, the project provided a large spectrum of contributions including **robot's navigation and localization** [88], telemedicine [167] and, mainly, **monitoring and context acquisition**.

Figure 3.5: CompabionAble project logo

User monitoring is a key feature for such a system and was accordingly addressed. Medjahed et al. [115] proposed a system to monitor the user health based on three types of sensors available in the environments and on fuzzy logic to ensure a proper data fusion. Activity monitoring was also tackled in CompanionAble. Volkhardt et al. [177] designed a system that identify the user's activity based on his/her current pose and motion, and on surrounding objects position. Situation awareness, and more generally context awareness, is explicitly addressed in this project. For instance, microphones are used for this matter [152]. Used robots are limited in capabilities; for instance, they have no arms, and they consequently performed relatively simple tasks, thus, task planning was not a main contribution of this work.

3.4 Fields of Problems for Personal Robots in Smart Homes

.

In these projects, we can identity five main fields of problems:

- **Network Communication**: the way the robot can interact with a smart environment from a technical point of view is a challenge. In fact, sensors are diverse and use different technologies, such as Bluetooth or Zigbee. They also have different ways to provide data: some provide events, some push the data, while other need the system to pull the data. Coping with this issue is already addressed in the literature [173, 168], including in PEIS[101, 73]. A solution is to rely on middleware, such as Universal Plug and Play (UPnP) [26] that abstracts network communication.

- **Localization and Mapping**: as they operate in homes, an essential challenge for personal robots is to map the environment and locate themselves. The current solution is to use Simultaneous Localization And Mapping (SLAM) [44, 17], however, not all personal robots have the sensors to use this technology. An alternative way is to rely on the smart environment. The possibility of using smart devices to locate robots was explored in multiple works and contexts. Some works explored the usage of laser range finders [58, 59], while others, including Robot-Era, tried to used RFID to locate robots [172, 90].

- **Human Robot Interaction**: personal robots are required to interact with users, and how they do so is an important matter, particularly considering a smart environment. In fact, smart devices

[9] `http://www.companionable.net`

can help the robot interact with the users, by easing their identification for example [166]. The robot can also be used as an interface for the smart environment [133] and the acceptance of such system by the user is being studied [166]. HRI is important to consider when proposing a personal robot, and is consequently commonly addressed [21, 60].

- **Context Acquisition and Management**: acquiring and understanding the knowledge of the context is essential. Actually, it allows the robot and the smart environment to take decision, intervene and act properly. This requires not only to gather context data from sensors, but also to understand them and enable a more abstracted knowledge. Although the robot can perform this process to some extent, in particular through vision algorithm [46, 62, 132], the smart environment can provide a greater knowledge. For this usage, RFID are commonly used [166, 30]. However, various sensors can also be used, sometimes very specifically, to monitor the health or activity of the user [152, 115, 177].

- **Task Planning**: finally, as mentioned, task planning allows the robot to compute a sequence of tasks to perform in order to reach a goal. Planning in consideration of the smart environment is important. Accordingly, some works, in particular in Robot-Era, proposed complex multi-agent planners [41], others consider the usage of probabilistic planner [87]. Nevertheless, the task planning for robots interacting with smart environments is not commonly addressed. However, out of smart environments, this topic is common for personal robots and many propositions solving issues including imperfect knowledge [185] or task failures [86] were studied.

Each field of problems has its own challenges and research community. Although there are all to be dealt with when conceiving and deploying a personal robot in a smart environment, **in this thesis, we focus on two aspects: context acquisition and management, and task planning**. In fact, as pointed out in the introduction, major open challenges are yet to be tackled in those fields. Yet, while existing approach of robots in smart environment do cover a large spectrum of problems, they also fail to overcome all challenges. Let us discuss those limitations.

Firstly, in order to act appropriately, robots need to acquire and understand the context. However, within a smart environment, sources of data are various, which implies heterogeneity over devices and context data. In the literature, the technical integration of sensors is commonly addressed by self-configuration, as in PEIS [73], or through middleware [26]. However, supporting the heterogeneity of context data is another challenge that is rarely explicitly addressed. On top of that, multiple approaches, including CompanionAble, use specific sensors and types of data, making these techniques dependent on these particular types of data. Having such a variety of devices induces problems of uncertainty. In fact, sensors can be imperfect, and sources can be in contradiction. This causes context data to possibly be outdated, inaccurate, imprecise, contradictory or incomplete. Considering the reliability of context data is important for the robot to properly understand the context. Yet, among works of robotics within smart environments, this issue is rarely addressed. In PEIS, it was partially considered [101] for anchorage. This leads to the open research question: how can a robot operating in a smart environment acquire the context over heterogeneous and uncertain data? Answering this question is essential as context understanding is closely related

to context acquisition. The use of the acquired context is impacted by issues of heterogeneity and uncertainty. The context knowledge can be used to understand situations, to infer complex data such as user activity, and to take decisions. This feature was implemented in CompanionAble, that proposed to monitor health dimensions [115] as well as the activities of the user [177]. However, they used specific sensors, as well as basic or raw context data with and focus on low level data fusion. More generally, reasoning over diverse and rich knowledge is often not addressed. Thus, how can personal robots understand the context and take decisions accordingly?

Secondly, once a decision was taken, the robot can achieve it using the acquired context knowledge and a task planner. Robot-Era did explore the task planning problem for robots in smart environments. In fact, as a part of this project, a whole task planner was designed [41]. This planner is able to deal with multiple agents and uses four solvers to ensure constraints of time, dependencies and resources. It allows to generate a plan with multiple devices without overusing them. However, the issue of task failures remains. Even if replanning is possible, it is not a satisfactory solution, yet this approach can't proactively avoid such failures. Furthermore, even if it prevents overusing devices thanks to its dedicated solver, the planner does not question the context acquisition. This leads to the usage of devices and context data that are not relevant in the plan making. Hence, the following problem remains: how can a personal robot avoid task failures? How does the robot task planner impact the smart environment?

We observed that the current research around personal robots in smart environment fails to answer those questions. However, these challenges are important to tackle to move toward high quality domestic robots. Yet, others communities have proposed techniques in other contexts that may be applied in our case. This is why, in the two next chapters, we review those techniques.

3.5 Conclusion

In this chapter, we offered a global view of problems encountered by personal robots in smart environments. Firstly, we described current personal robots to understand their capabilities and limits. Smart home can offer a way to help these robots, consequently, multiple major projects were conducted. These projects aimed to use robots in domestic context and used environment enriched with sensors and actuators. To achieve that they had to cope with a large spectrum of problems. We pointed out five main fields of problems that are to be tackled for a personal robot in smart home. As these problems overlap various domain and competencies, in this work, we focus on two fields of problems: task planning and context management. Among existing work of robots in smart environments, we pinpointed limits in these two fields. Accordingly, we will now review the literature of both context acquisition and management and task planning techniques in the next two chapters respectively. From these analyses, we aim to evaluate what state-of-the-art techniques can be applied and what are their limitations.

Chapter 4

Context Awareness

4.1 Introduction

Acquiring, understanding, and using the context knowledge is an essential step for a personal robot and/or a smart environment. This knowledge is the key to their intelligence. Indeed, it is not only useful, but also required, to make decisions and act in the environment. In fact, task planners are dependent on the context knowledge and cannot provide a task plan without it. However, dealing with the context is not trivial, in particular in domestic applications. As we mentioned, there are numerous challenges, including the heterogeneity of data, the variability of sources, uncertainty, and the ability to identify particular situations. In the previous chapter, we pointed out that most works addressing robots in smart environments do not provide answers to these challenges. Consequently, in this chapter, we aim to review more general methods proposed in the literature and used to acquire, understand and use the context. These techniques were used in various applications [1], in particular smart homes and other smart environments, but were rarely considering robots. Nevertheless, we analysed them in regards to our challenges.

The general process of context management is not only to acquire sensor data, but also to rise up in abstraction. The idea is to increase the semantics of data toward information and knowledge that can then be used for accurate decision making. As depicted in Figure 4.1, we can decompose the context management into four layers. Firstly, sensors and other sources provide *context data* that represent a basic dimension in the environment, for example the detection of a movement is a *context data*. Secondly, these context data can be aggregated to form higher level *information*. Typically, the user location that is inferred from detected movements, sounds and others, is an *information*. Thirdly, with all these context data and information, the robot and/or smart environment can understand the global situation of the environment, for example it enables Nono to understand the anomalous situation when Arthur forgets he is cooking. Lastly, based on this knowledge, the robot can decide what to do in response, in Nono's case, it decides to go and turn off the oven. This whole abstraction process is required for a robot in a smart environment, but it is enabled through multiple phases. In fact, in the literature, approaches often deal with only some parts of this process. Some works are dedicated to provide particular information from context data, such as user activity, while others use already acquired information to infer new information or understand the

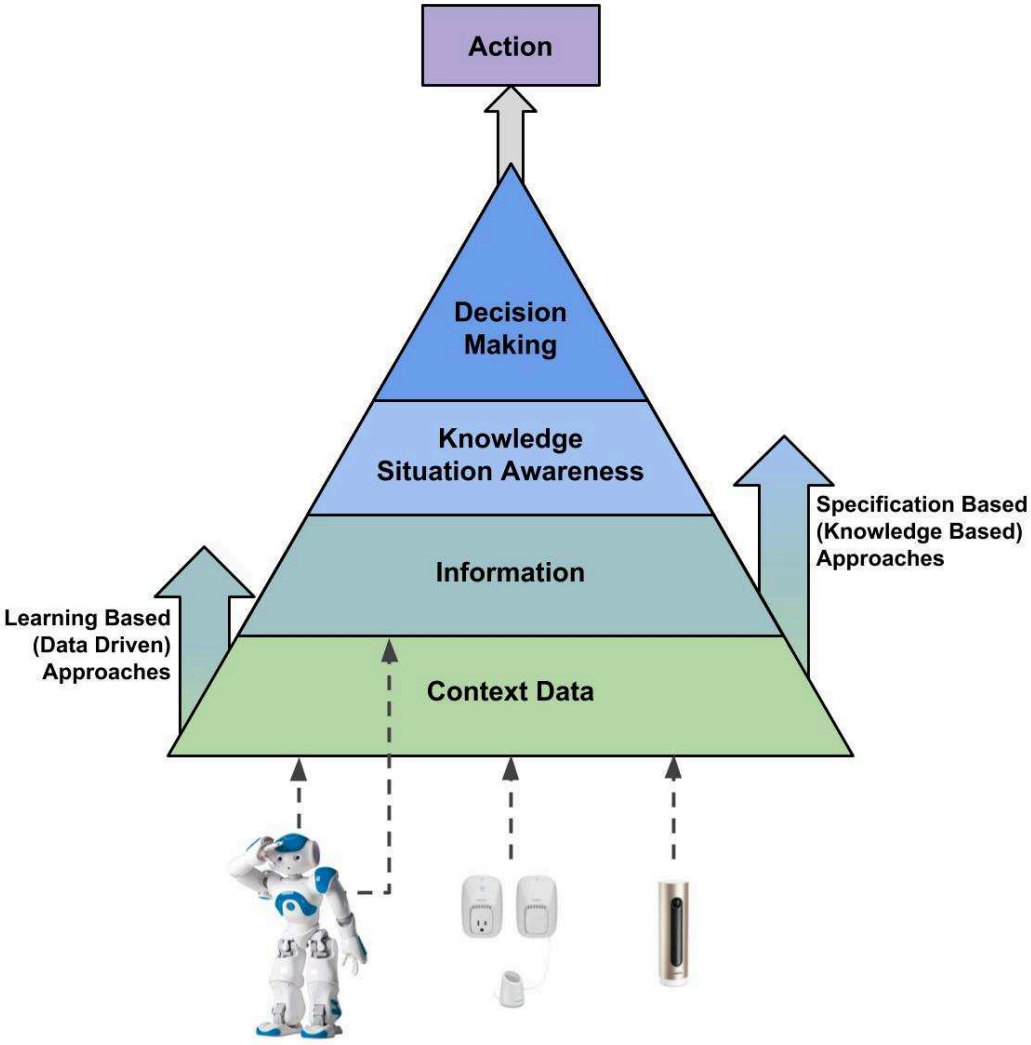


Figure 4.1: Data Abstraction Process

situation. Thus, a complete context management solution is likely to use multiple tools to acquire, understand and use the context. The literature carries multiple approaches and vocabularies, but before reviewing those works, let us discuss the terms and notions implied.

4.2 Background and Definitions

Contributions around context management are diverse in paradigms and applications. The terms used in the literature can have different definitions from one work to another. In this section, we aim to provide definitions of notions used in this chapter. Furthermore, we also discuss the position and classification of state-of-the-art solutions.

The very first notion to address is naturally the *context data* or *contextual data*. *Context data* are the first to be acquired and can then be used to infer *information*. We consider the following definition of context data:

Definition 1. ***Context Data (or Contextual Data[1]):***
A context data is a formatted data that represents a specific dimension of the environment.

For example, the temperature in a room, the detection of a movement or the robot battery level are *context data*. Such data are provided by sensors or other data sources, and are formatted according to the technique used. In this work we focus on the management of data, how the raw data provided by sensors are technically acquired and transformed is not in the scope of this thesis. Indeed, we do suppose the usage of an interface or a middleware in charge of communicating with the devices. Nevertheless, it is important to notice that the *context data* can be provided in different ways. Some are provided as events, while some are pulled or pushed to the system. Solutions in the literature can be different according to the hypotheses used as for data sources. The acquisition of the context data is the first layer of the abstraction of the context, as seen in Figure 4.1.

These context data can be abstracted into *information*. *Information* is a higher level data that is inferred from the analysis of context data, or other information.

Definition 2. ***Information:***
A piece of information is a high level data inferred from derivation of one or multiple context data.

For instance, the user location is an *information* derived from context data representing the movement or noise in rooms. *Information* is obtained from other context data, but can also be provided by smart devices or robots that perform a first analysis by themselves. For example, the location of the user may be directly provided by the robot using a vision algorithm. One particular *information* is the user's *activity*. Recognizing the *activity* of the user is a popular topic in the literature. In fact, multiple contributions are proposed to solely extract this information. It is important to distinguish *activities* and *situations*: an *activity* refers to the user's actions, while the situation is a global state of the environment. In the literature, these two terms are often used ambiguously. Hence, we define the activity as follows:

Definition 3. ***Activity:***
An activity is a piece of information related to a given user and that describes his/her action.

Arthur cooking is an example of an *activity*. Note that an *activity* is exclusively related to a human user. As for the robots, we use the term *task*. Many works aim to extract the activity by itself, for instance to provide it to medical staff, but the activity is also a precious information for understanding *situations*.

The *situations* are a more global concept that are higher in abstraction. In fact, they can be seen as a snapshot of the environment and is based on a large set of information and context data, including activities.

Definition 4. ***Situation:***
A situation represents a given state of the environment.

[1] In the literature, the two terms are used in a similar way [189]: both are correct and can be applied, nevertheless, in this document we will use the term *context data.*

Understanding situations means knowing what is happening in the context. It abstracts a large amount of *information* and *context data* into knowledge, as represented in Figure 4.1. For example, when the oven is active while Arthur is elsewhere sleeping, the *situation* is abnormal and should be solved. Similarly to humans, situation awareness [47] is essential for robots and/or smart devices for decision making. With this knowledge, the system can decide what to perform. This usually implies calling a task planner, thus ending the context management process.

In the literature, various definitions of context data, information, activities and situations can be found. In this work, we will rely on the ones provided here. Nevertheless, even if not exactly using the same terms, state-of-the-art works do aim to acquire and understand the context. To this extent, various propositions were conducted to acquire higher level knowledge or, in other words, to rise up the abstraction of data. Many approaches cope with a small part of this abstraction. Some may focus on inferring particular information, while others may tackle specifically situation awareness from information. Most of them are actually addressed in regard of activity recognition and/or situation awareness. According to the INCOME project[2] [8], these techniques can be classified into two main categories: data driven approaches and knowledge driven ones. **Data driven** approaches usually focus on using low level context data to infer new information, such as activities. Hence, these techniques allow to obtain information from context data, as depicted in Figure 4.1. Accordingly, they often rely on machine learning using basic data. On the other hand, **knowledge-based** approaches rely on specified knowledge, through rules for instance, to perform reasoning in order to generate information or situations. Such techniques can model and use information to infer complex situations or activities. Consequently, they often permit to rise from information to knowledge, even if they can be used to infer new information as well, as seen in Figure 4.1. These approaches can also be similarly divided as learning-based and specification-based methods [190].

In the next sections, we will review the techniques used for abstracting data and creating context knowledge. Most of this analyses were mostly based on the reviews of surveys conducted by Ye et al. [190, 189]. We will review each category of techniques and evaluate their strengths and weaknesses, in particular facing our challenges. Consequently, we will first focus on learning-based approaches, before tackling specification-based ones. Then, we discuss the possibility of using both learning and specification through **hybrid approaches**.

4.3 Learning-Based Approaches

Learning-based techniques rely on a dataset to learn a model from it. Once the model is learnt, it can be used to classify, in our case, context data into activities or situations. By doing so, they do not require a specification effort, but the learning phase is critical. Some models, once settled, cannot evolve, while other can continuously evolve as the environment does so. Using machine learning was commonly utilized for activity recognition or situation awareness. Let us review the most important techniques.

First, a simple approach is to use the Bayes theorem: this is the method followed by Naive Bayes or Bayesian Networks [121]. It estimates the probability of the hypotheses by knowing the proba-

[2] `www.irit.fr/income/`

bility associated to evidences when the hypotheses is true thanks to the learning set. Although it can achieve decent results, it requires an accurate supervised learning phase and compatible data. Moreover, the model can hardly evolve at runtime.

A more mature usage of Bayes theory is the Bayesian Network (BN) [53]. A BN is a acyclic graph where nodes are variables and arcs are relations between variables: if an arc is represented from A to B, it means B depends on the variable A. Such a model allows to represent the dependencies between context data, activities and/or situations. Classification is done by propagating probabilities using a probability table associated to the nodes. They were used in multiple works in context aware systems [175]. BN main issue is its weakness toward incompleteness: BN does not support well missing data. Furthermore, even if it is a supervised learning-based technique, it requires some expert interventions, in particular for the structure of the graph. Once the model is set, it does not change without expert intervention.

Hidden Markov Model (HMM) is also a common solution for classification [23]. Such model relies on states and observations, which, in activity classification, can be associated respectively to activities and context data, as depicted in Figure 4.2. The principle is to estimate the probability of moving from a state to another according to the observation. They are perfectly suitable for context awareness [103], yet they are sensible to scalability, as the number of states can be important. HMM uses supervised learning.

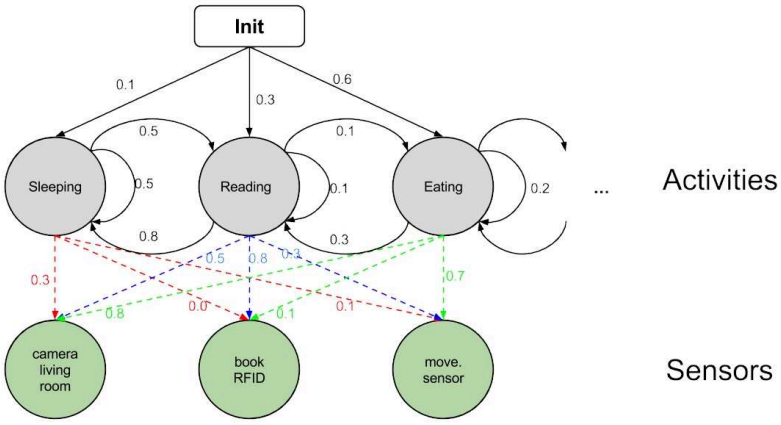


Figure 4.2: Example of HMM

States are grey nodes while observations are green nodes. An arc represents the probability of going from a state to another, and the probability of having a given observation when entering the state.

HMM supposes all observations are independent, which prevents understanding long-term and complex patterns. Conditional Random Field (CRF) [97] aims to solve that issue by considering sequences of states and not only transitions between states. CRF can be seen as an undirected acyclic graph. Similarly to HMM, the weights are learnt from a dataset under supervision, while the feature functions and the structure are provided by an expert. They are commonly used in vision, but can also be used for activity identification [174].

Support Vector Machine (SVM) is yet another machine learning-based approach [72] that can be used for context awareness. SVM principle relies on finding the optimal linear hyperplane separation between classes in the training dataset. SVM are able to operate with various number of features, but are sensible to uncertainty. They were used for activity recognition and gesture classification [6].

Context Free Grammar (CFG) [119] relies on a decomposition knowledge of situations into actions, and actions into sub-actions. Detection is performed by detecting pattern of actions through production rules. A variation of CFG is Stochastic Context Free Grammar (SCFG). In SCFG, production rules are associated with probabilities that are learnt from a dataset. Although CFG and SCFG are suitable for basic scenarios with simple and known structure of interest, they are not adapted for more complex case as the knowledge required would need a huge effort from the expert. Learning the structure of CFG is a way to explore, but it was shown to be difficult.

Decision Tree (DT) is a very common technique in machine learning [154]. A DT represents classes as a tree were nodes represent criteria of classification over features. It uses this formalism to classify entities based on their features. DT are learnt without supervision from a dataset by extracting features that allow to separate classes. They have been used activity recognition from sensors, but can only operate on small and simple dataset.

Another approach is to use web mining to extract knowledge from distant databases when datasets are not available [36]. For instance, by consulting a "howto" website, it is possible to determine what objects are used during a particular activity. The knowledge can supposedly be used at any time, including when deployed. Although it was applied in home context [188], it is very specific and can only solve particular cases.

Although initially designed for text analysis, suffix trees were used in activity recognition [117, 69]. It consists of discovering temporal patterns of events to identify activities, that are considered to be sequence of events. In fact, a suffix tree represents all possible sequences and sub-sequences of events for a given activity, as depicted in Figure 4.3. The events pattern, thus the structure of the tree, are learnt. It is to be noted that, by basing on events pattern, suffix tree are limited to event oriented data.

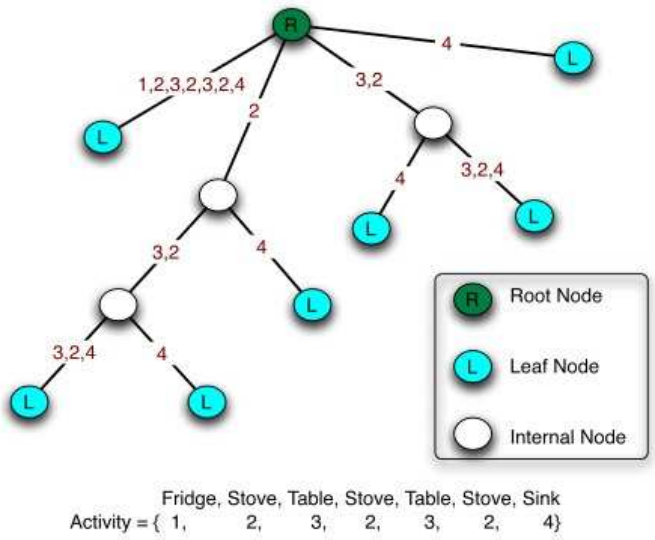


Figure 4.3: Example of suffix tree representation for an activity from [69, 190]

Finally, Artificial Neural Networks (ANN) simulate the behaviour of a layer based network of neurons. A neuron has weighted numerical inputs and is activated if the sum of its input goes higher than a threshold. Outputs of neurons from one layer are linked to the inputs of neurons of the next layer. For activity recognition, the neural network uses as inputs, meaning inputs for the neurons of the first layer, sensor data. The network is then interpreted and the output is a selection of an activity among a list. Figure 4.4 depicts an example of an ANN. ANN weights are usually learnt and adapted on the fly, for instance by using genetic algorithms. However, ANN mainly works as a black box, meaning it is hard to adjust if it is mistaken. Furthermore, uncertainty can have a huge impact on the recognition process.

Learning-based approaches characteristics are summarized in Table 4.1.

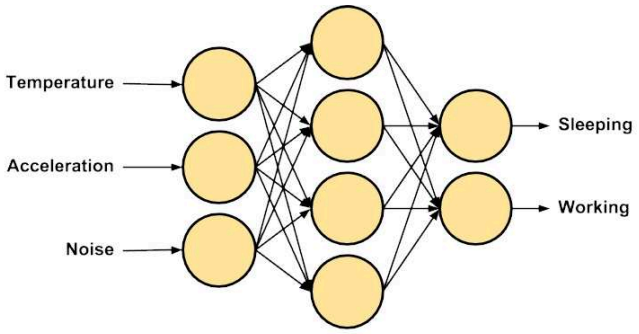


Figure 4.4: Simple example of ANN

This ANN is composed of 3 layers of artificial neurons, represented as yellow circles. Based on some context data, the ANN is able to detect if the activity sleeping and/or working are occurring.

4.4 Specification-Based Approaches

Opposite to learning techniques, approaches relying on specification are also popular for context awareness. Specification requires a greater engineering effort to set, as an expert must provide relatively detailed information, but it enables a refined and more high-level recognition process. In fact, many machine learning approaches only use simple data and can cope with basic situations. Specification-based approaches on the other hand, can detect potentially complex activities and situations. Let us review some of these techniques.

First of all, the most general approach consists of using formal logic, through logic programming [74]. Logic programming consists of specifying rules that define high-level data, including situations or activities. These rules are grouped in a rule base. Multiple formalisms are possible for these rules. As situations cannot be inferred from raw data, in most cases, backward or forward chaining is required to go from sensor data to situations/activities. Formal logic based situation aware systems have the advantage of having a formal specification of situations, as well as enabling verification of consistency of these specifications. On top of that, as it relies on rules, it is easy to integrate new sensors and new data by adding rules to the rule base. Completeness issue is supported as multiple 'ways', through different rules, can be found to evaluate the activity. However, other uncertainty

	Type	Uncertainty	Heterogeneity	Evolution
Naive Bayes	Supervised	Inaccuracy Contradiction	No	No
BN	Supervised	Inaccuracy Contradiction	Yes	No
HMM	Supervised	Inaccuracy Contradiction	Yes	No
SVM	Supervised	NA	Yes	No
CRF	Supervised	Inaccuracy Contradiction Freshness	Yes	No
CFG	Supervised	Completeness Inaccuracy Contradiction	Yes	No
DT	Supervised	Completeness Inaccuracy Contradiction	Yes	No
Web Mining	Unsupervised	NA	Yes	Yes
Suffix Tree	Unsupervised	NA	No	No
ANN	Unsupervised	Completeness	Yes	Yes

Table 4.1: Comparison of learning-based context awareness techniques

dimensions are not supported.

Formal logic cannot handle efficiently space and time properties. That is why Spatio-Temporal Logic was proposed [11]. This logic extends the formal logic with dedicated operators such as 'AND-Later'. Some works are even able to handle trajectories, which is convenient for robotic applications. Yet, such a logic requires a particular data structure and solely focuses on spatio-temporal data.

A very popular logic that copes with uncertainty is the fuzzy logic [93]. Fuzzy logic relies on partial truth. In fact, a statement is associated with a truth degree and can overlap with other statements. As a matter of fact, in fuzzy set theory, a variable can belong to multiple sets at the same time with various truth degrees. The most common example is the temperature: 10°C can be considered cold, 20°C considered hot, and 15°C considered both cold, with truth degree 0.5, and hot with truth degree 0.5. Using such a logic in context aware systems allows to cope with the imprecision of context data, as well as contradictions. It is commonly used [115], but rarely by itself.

Dempster-Shafer Theory (DST) is a mathematical theory of evidence proposed by Dempster and Shafer [164]. It has been applied to recognize activity in pervasive environments [114]. Its main strength is its ability to propagate uncertainty values, allowing to provide an information on the certainty of inference. DST is a graphical model that relies on mass functions, frame of discernment and combination rules. It relies on an oriented networks of proofs: sensors are represented as input while activities are positioned as outputs, in between, we can find intermediate context data as well

as operations. DST uses mass functions on data: each subset of possible values of data is associated to a mass, that is a degree of belief. This belief is distributed in the network using specific operators and combination rules, such as composition, derivation or association, as depicted in Figure 4.5. The belief network and the degree of belief has to be specified by an expert. These degrees of belief are propagated in the network according to the sensory observation, allowing to cope with completeness and contradiction.

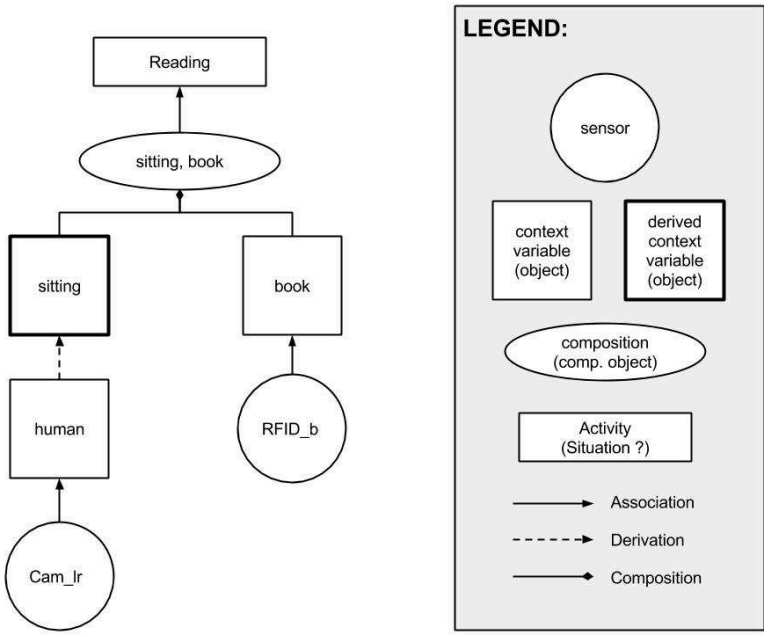


Figure 4.5: Example of DST.

Situation theory is a variation of logic [112]. Situation theory, or situation calculus, is another form of logic particularly interesting in our case. In this logic, a situation is defined as a state of the world at a given time. Such a logic is designed to deal with actions leading to situation changes. Kalyan et al proposed a multi-level situation theory [84]. This upgraded approach relies on infons, a discrete unit of information for a single entity, and situations. Infons are obtained from sensors or reasoners. Situations are defined as a set of sub-situations, or micro-situations, that are themselves defined as a composition of infons. Consequently, by creating entity-specific micro situation and then combining them, we can determine the current situation.

Last but not least, ontologies are also a powerful tool for context awareness. Ontologies were initially designed and used for the Semantic Web [3], yet, they have now more applications. One of the most popular ontologies representation is Resource Description Framework (RDF), that relies on RDF triples: subject, that is the resource to describe, predicate, that is the property applicable to the subject, and object, that is the value of the property and can be another resource. Hence, an ontology can be seen as a set of RDF triples[4] and can be seen as a graph with resources as nodes and properties between them as arcs. Such a formalism allows for a clear and rich representation of data. Furthermore, ontologies can represent a lot of different data, from sensor outputs to user activities. Ontologies can rely on various language definitions, such as OWL, that enable high interoperability

[3] `https://www.w3.org/standards/semanticweb/`

[4] Note that other ontology representations exist, yet, in this work, we focused on the RDF formalism.

between systems. One of the major strengths of ontologies is their ability to support rule-based reasoning engines, allowing to infer higher level data. Finally, ontologies can be easily extended with new concepts, reasoning engine or other features. However, on the other hand, they require a significant specification effort to design the structure and the rules. For these reasons, they are extensively used for situation aware applications [148] and often combined with other techniques, as we will see in the next section.

Characteristics of specification-based approach can be found in Table 4.2.

	Uncertainty	**Heterogeneity**	**Evolution**
Formal Logic	Completeness	Yes	Addition of rules
ST Logic	Freshness	No	Addition of rules
Fuzzy Logic	Imprecision Contradiction	Yes	Addition of rules
DST	Completeness Contradiction	No	No
Situation Th.	NA	Yes	Addition of rules
Ontology	Via extension	Yes	Addition of rules

Table 4.2: Comparison of specification-based context awareness techniques

4.5 A Need For Combination

In the previous two subsections, we reviewed numerous techniques from various fields. We can note major differences between learning-based and specification-based approaches. Learning-based approaches, as summarized in Table 4.1, are noticeably adapted for uncertainty. In fact, several techniques are able to solve several uncertainty dimensions at a time, such as CRF, CFG and BT. They are also in most cases suitable for heterogeneous data. It is to notice that unsupervised learning approaches are not doing well with uncertainty. However, learning-based approaches often deal with simple and static models, that require a new learning phase if the environment changes. Furthermore, learning only allows to understand the usual situations and activities that are presented in the training dataset. In consequence, learning-based approaches can hardly cope with non-recurrent situations. On the other hand, specification-based approaches are neither very well adapted to uncertainty management nor are designed to only solve particular uncertainty dimensions , such as Fuzzy Logic or DST. However, they are able to cope with complex situations and can be extended by adding new rules, even if it requires further engineering effort.

As we can see, both types of approaches partially support our constraints, but they appear to be complementary. This leads to the idea of hybrid approaches, as pointed by Ye et al. [190]. These are solutions that combine several techniques, both learning-based and specification-based. Such approaches are specifically designed for context awareness in home environment, hence, they would allow to cover more constraints. Furthermore, such approach can cover a larger abstraction of data, by not only being able to infer information, but also enabling a complete knowledge through sit-

uations. It is to notice that some of these approaches are actually more relevant when combined, for instance, web mining seems to have a greater potential when combined as a complement with another technique. Ontologies are also perfectly suitable for combination, as they could be easily extended. Many combinations are actually possible. In the literature, multiple hybrid approaches were proposed, in particular based on ontologies. Each provides features and solves particular issues. Yet, all of them aim to have the most accurate understanding of the context. This can be measured by the activity or situation recognition accuracy for each approach. This metric is a good way to compare approaches, even if hypotheses vary from one to another, for example, some techniques may be designed to overcome one particular uncertainty dimension and will struggle in an environment subjected to other dimensions. Let us review hybrid and ontological approaches.

4.6 Combination Approaches

Hybrid approaches, that are combinations of techniques, were proposed in order to take advantage of the strengths of both types of approaches while limiting their weaknesses. Such a way could lead to an efficient context aware system in smart home, that is why it was explored by multiple works that we review in this section.

Multi-Entity Bayesian Network (MEBN) [99] is a typical hybrid approach. MEBN is an upgrade of Bayesian Networks based on first order logic and Bayesian theory. In fact, on the one hand, a BN lacks expressiveness and considers a finite set of entities. On the other hand, formal logic does not have a proper way to deal with uncertainty. By using both formalisms, MEBN enables a high expressiveness while supporting uncertainty. MEBN relies on 'fragments', that can be seen as small BN. Each fragment carries nodes that can be interfaced with other fragments. Based on the context observations, fragments are instantiated and aggregated, the outcome results in a situation-specific Bayesian Network, that is a BN perfectly adapted to the current context. Thus, MEBN is actually able to handle the evolution of the environment: it dynamically creates a situation-specific Bayesian Network and the theory can be easily enriched with new fragments. On top of that, it supports the same uncertainty dimensions as Bayes based approaches, namely inaccuracy and contradiction. However, they cannot cope with other dimensions and can not use reasoning for a richer context knowledge. Later, MEBN was used to extend the OWL ontology language through PR-OWL[5] [39]. PR-OWL aims to add probabilistic representation into OWL for a better uncertainty management. However, although this hybrid approach stood out in space encounters scenarios[6], it was not applied for domestic context awareness. In the end, PR-OWL features the expressibility, interoperability and reasoning of ontologies, as well as a support of inaccuracy and contradiction. It means that completeness, precision and freshness are not tackled.

First-order logic allows to model rich and complex knowledge from various context data. However, it does not support uncertainty, unlike probabilist Markovian approaches. That is why Richardson and Domingos proposed [149] a combination called Markov Logic Network (MLN). A Markov logic network is a first-order knowledge base with a weight attached to each formula or rule. It can

[5] www.pr-owl.org/

[6] www.pr-owl.org/basics/ontostartrek.php

be seen as a template for constructing Markov networks. Those networks can then be used to evaluate the validity of the formula. MLN takes the best of the two approaches and models complex knowledge while supporting contradiction, imprecision and completeness. However, MLN does not support the other uncertainty dimensions. Furthermore, it does not rely on semantic knowledge, however it can be associated with ontologies.

Considering fuzzy property is an interesting perspective. That is why some studies explore the usage of both fuzzy logic and ontologies. Rodriguez et al. [151] proposed such a solution. In their work, the authors proposed a fuzzy ontology to represent human activities. To do so, they used Fuzzy OWL 2, an extension of the OWL language, as well as FuzzyDL, a fuzzy reasoning engine. They designed an ontology based on crisp ontologies [24] and concepts extracted from other ontologies such as CONON [182] or CoDAMoS [138]. This resulting model is composed of four main entities: users, environment, activities and relationship, as shown in Figure 4.6. This approach was compared to other crisp ontologies in terms of accuracy and scalability . Thanks to its fuzzy properties, it has proven to achieve better accuracy for an equivalent scalability. However, in our context, scalability is not a main matter as we only consider home environments. Moreover, fuzzy logic allows to tackle uncertainty to a certain extent, but cannot cope with other dimensions such as completeness or contradiction.

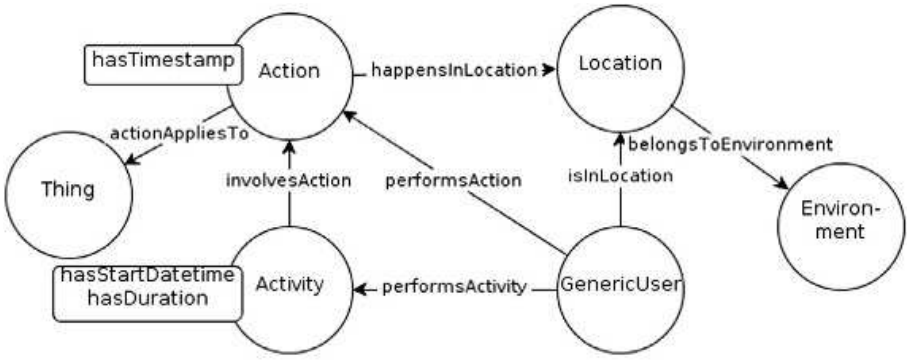


Figure 4.6: Main relationships from the context ontology [151]

Relationships between User, Activity and Environment, from [151]

Another work using fuzzy logic is the proposition of D'Aniello et al. [45]. In their work, the authors aim to provide a multi-agent fuzzy-based approach for situation detection. To do so, they use three ontologies: Semantic Sensor Network Ontology (SSNO) [35] to describe the smart environment, Situation Awareness ontology to describe the situations, and domain specific ontologies such as time ontology or geonames ontology. The three ontologies can be seen in the architecture schema presented in Figure 4.7. They used a multi-agent and multi-layer approach to enable a better interface with the smart environment. Furthermore, to classify data, they relied on Fuzzy Format Concept Analysis (FFCA) as well as fuzzy association rules. This enabled the use of the strength of fuzzy logic to their advantage and cope with imprecision of data. On top of that, by using association rules mining, completeness is supported. This work has serious strengths for home environments context awareness, yet it does not overcome all uncertainty dimensions.

Attard el al. [10] describes a novel context ontology: DCON. It is illustrated in Figure 4.8. DCON

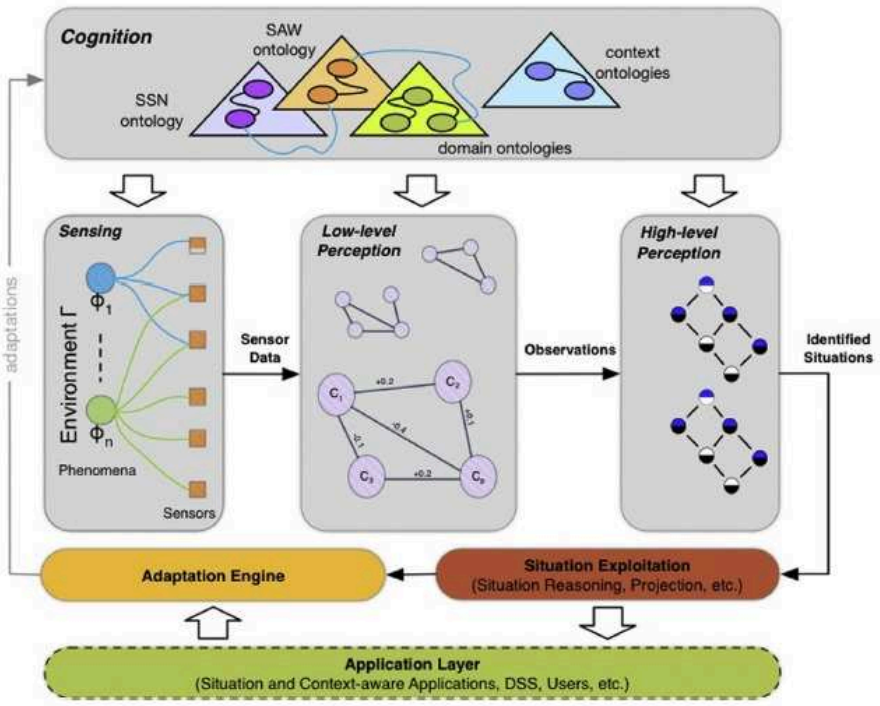


Figure 4.7: Architecture of the hybrid approach proposed in [45]

actually describes a 'snapshot' of the state of the environment. A situation is defined as a given context, thus a situation is described by one DCON ontology. The idea of this approach is to maintain a live context and to compare it to stored situations. DCON relies on layers: attributes (which are raw values), elements, aspects and context. It is defined by three sub-ontologies which may be seen as three different model levels, where each level defines the next one. As said before, situation recognition is performed by comparing the live context to stored situations by similarity measurement. In stored DCON, each aspect, element and attribute carries a weight that indicates the relevance of an information for this situation. This weight is then used in the similarity algorithm. With this weight notion, DCON is able to take into account users feedback by adjusting weights according to positive or negative examples. This is a very interesting feature for a personal robot: user satisfaction is not to be neglected. However, the approach does not support uncertainty.

Finally, Okeyo et al. [128] proposed a way to combine ontology and temporal formalism for activity recognition. In their work, activities are decomposed by sub-activities. Sub-activities have dependencies between each other, for instance, an activity can follow another. An ontological model was defined: Activity of Daily Life (ADL): ADL allows to describe sensors, objects, context and more. Unlike DCON that represents 'snapshots', ADL contains all known activities. The core of this work is its window oriented data management. In fact, the system is able to support continuous data stream and to cope with freshness issues. For activity recognition, the solution uses four steps. First, it transforms context observations into basic actions by using the ADL ontology. Secondly, it generates partial activity representation, using the ontology again. From this, the system then applies simple activity recognition by using subsumption and similarity measurement. Finally, complex activities are recognized by applying rules. This process is not very detailed as the core of this work is the temporal aspect. Here, the ontology is mainly used for storing the knowledge of activities rather

than storing the context itself. Furthermore, only freshness is supported.

The features of each reviewed hybrid approach is depicted in Table 4.3. As we can see, all reviewed approaches can model heterogeneous types of data. Furthermore, similarly to specification-based, they rely on rules, that can be added and/or adapted to the environment. It is worth noticing that ontologies are widely used, enabling semantic knowledge, inference and interoperability. However, uncertainty is variably supported. The approach proposed by D'Aniello et al. [45] is the most advanced to overcome these challenges. They provide a full abstraction of context data into situations and cope with three uncertainty dimension thanks to fuzzy logic and association rules mining. Yet, they do not tackle neither freshness nor contradiction. Hence, overcoming all uncertainty dimensions is an open research problem. Further combinations are to be explored to get closer to a full uncertainty support.

	Semantic	Uncertainty	Heterogeneity	Evolution
MEBN [99]	No	Inaccuracy Contradiction	Yes	No
PR-OWL [39]	Yes	Inaccuracy Contradiction	Yes	No
MLN [149]	No	Contradiction Imprecision Completeness	Yes	Addition of rules
[151]	Yes	Inaccuracy Imprecision	Yes	Addition of rules
[45]	Yes	Inaccuracy Imprecision Completeness	Yes	Addition of rules
[10]	Yes	No	Yes	Addition of rules
[128]	Yes	Freshness	Yes	Addition of rules

Table 4.3: Comparison table of all compared hybrid approaches

4.7 Conclusion

In this chapter, we reviewed context aware techniques used for activity recognition and/or situation awareness, in particular in smart homes applications. We discussed and criticized learning-based and specification-based approaches with focus on our objectives. We noticed that each type of approaches was able to solve some of our constraints, but not all of them. On the one hand, learning-based methods tend to be more efficient as for uncertainty management. On the other hand, specification-based methods can be easily enhanced by adding new rules and can detect complex patterns. From this observation, comes the idea of using combination of learning-based and specification-based approaches: hybrid solutions.

Hybrid approaches are promising and were consequently explored for domestic context aware-

ness by multiple researchers. The state-of-the-art points out the relevance and performance of such approaches. In fact, by using the best of both learning-based and specification-based methods, such approaches are achieving better results for activity recognition and/or situation awareness. As summarized in Table 4.3, it is also worth noticing that most of them rely on ontological knowledge: this enables interoperability, reasoning and a high-level data modelling. This partly explains why all hybrid approaches can model and use heterogeneous types of context data. Most of them can be extended by adding new rules, by opposition to learning approaches that need to recreate the model. However, uncertainty is only partially covered. At best, three uncertainty dimensions are supported [149, 45]. Hence, coping with our five identified uncertainty dimensions is an open problem. Another finding is that many combinations were not tried. Thus, exploring new hybrid approaches is a promising way toward handling of uncertainty.

In regard to the observed literature limitations, we explored new combinations through three contributions. Firstly, we proposed two context acquisition techniques for both event-based sources and video-based ones, such as robot cameras. We designed a technique that gathers, batches, semantizes and fuzzifies events. This approach actually combines CEP with ontologies and fuzzy logic. It is able to model and/or cope with four uncertainty dimensions that are freshness, accuracy, precision and contradiction. We also improved a learning-based vision algorithm for activity recognition. We actually refined the vision results with an ontological knowledge to enable more precise and accurate detections. Secondly, based the acquired knowledge, we proposed CAREDAS, a technique that identifies anomalous situations by using Markov logic and ontologies. This situation awareness then allows to make a decision. Our contribution is able to take into consideration the previously modelled uncertainty and supports completeness. When operating together, these contributions are able to acquire and use the context over uncertainty and heterogeneity. Furthermore, they can be adapted to the environment through the specification of rules. All in all, our contributions allow the robot, helped by the smart environments, to have a rich context knowledge and to take an appropriate decision. They are presented in Part II. Once the context is acquired and decision is taken, the robot can rely on a task planner to reach its goal, which is also subjected to challenges, as we will see in the next chapter.

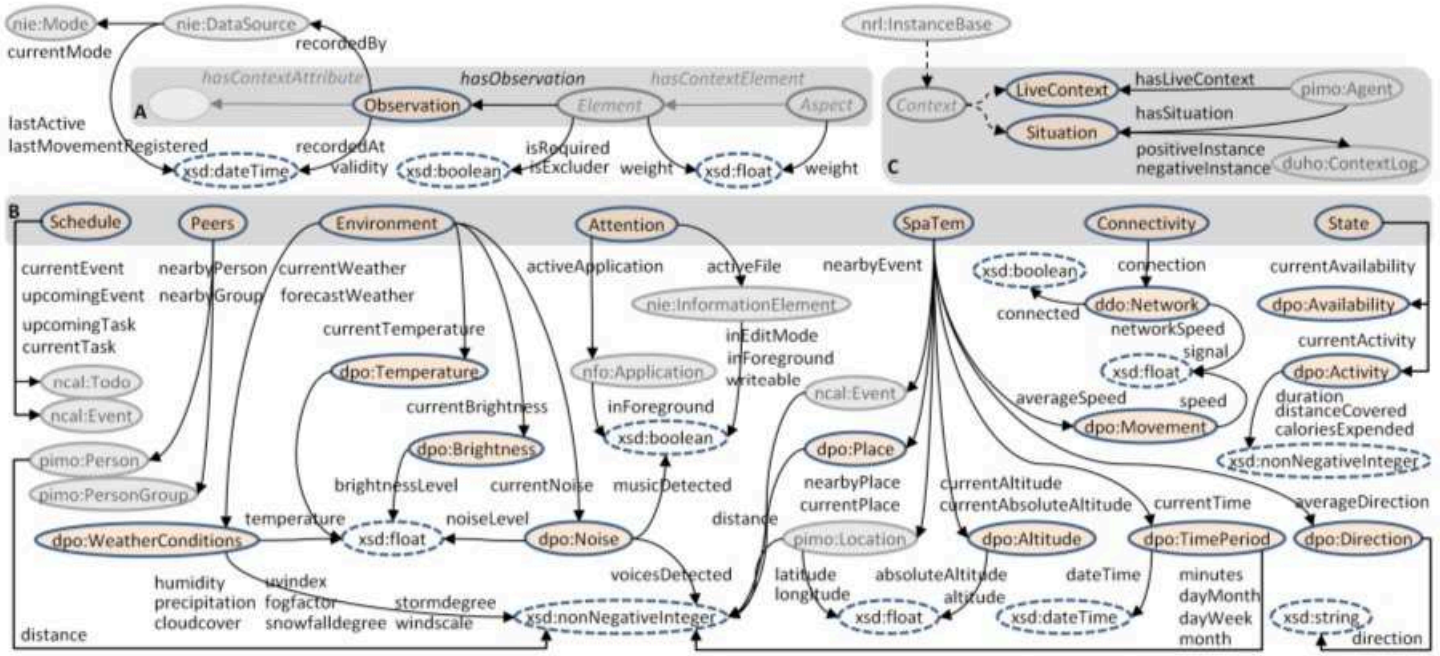


Figure 4.8: DCON ontology from [10]

Chapter 5

Automated Planning Techniques

5.1 Introduction

One of the important features for personal robots is the control of their actions and behaviours in the environment. Indeed, once it understands its surroundings, a robot must act. But to determine what to do and how to do it, it uses a control system or a task planner. As we previously pointed out, although it is a common problem in robotics, with multiple solutions, it yet lacks exploration for robots in smart environments; in fact most planning approaches focuses on standalone robots or other fields of application.

There exist numerous techniques for controlling robots. One approach is to consider all possible states of the robot and the environment. Several models explored that direction and were used in robotics. Among these techniques, we can find Finite State Machine (FSM) and Hierarchical Finite State Machine (HFSM) [33, 95, 77], Behaviour Tree (BT) [111], and Influence Diagram (ID) [96]. However, such approaches are only suitable for simple cases: indeed, in real case scenarios, foreseeing all possible cases of a personal robot in a smart home is almost impossible. Furthermore, FSM and BT are sensible to the problem of maintenance and scalability. Finally, even if the model is exhaustive, upgrading it to environmental changes requires a large engineering effort, as relationships between all other states are to be modelled. For these reasons, they are not suitable for domestic service robots. Another approach consists of using Artificial Neural Networks (ANN) to control the robot [16, 137]. In that case, the ANN uses as input context data and provides as output a task to perform. Although it can work with a limited expert intervention, it requires an exhaustive and accurate training phase. Moreover, as ANN work as a "black box", they do lack control; this can be a problem as the robot may take inappropriate decisions, in particular in unplanned situations.

Another way to control robots is through task planning. The principle is to generate a sequence of tasks according to the current context by using the knowledge on the tasks provided by an expert. By doing so, the planner features flexibility and control at the same time. Automated task planning is a large topic with multiple applications, including robotics. Even if commonly used for personal robots, task planners are still encountering limits in domestic applications and smart environments. In this section, we review the state-of-the-art of planners for personal robots. First of all, before addressing planning for robots, we review the general class of task planner. We then identify what

are the constraints and specific challenges for planning in a domestic robotic context. Finally, we review the literature around robot task planner before discussing them toward defined criteria.

5.2 General Task Planning

Automated planning is a research topic on its own that was studied for more than 40 years. It has numerous usages, nevertheless, AI and robotics are among its core applications. There exist multiple planning solutions each having its own characteristics in terms of definitions and algorithms.

5.2.1 Chain Planner

Stanford Research Institute Problem Solver (STRIPS) [51] is one the most famous and pioneer planners. STRIPS considers a set of possible actions, each having preconditions and postconditions. The knowledge of the 'world' is represented through predicates. Actions conditions respectively represent the required predicates to perform the action and the effect of the action on the "world". Given the current state of the "world", and a goal state to reach, STRIPS generates a sequence of actions that lead to the goal state. To do so, it chains actions according to their preconditions and postconditions using a stack-based algorithm. An example can be found in Figure 5.1. The formalism and the method proposed by Fikes and Nilsson was intensively used over the years. Particularly, STRIPS formalism lead to Action Description Language (ADL) [135], and later to the famous Planning Domain Description Language (PDDL) [113]. STRIPS was improved, adapted and led to new planning solutions according to the needs and contexts. For example, some improved and used it for video games by using a A* algorithm [129], while others proposed a planner relying on constraints [176].

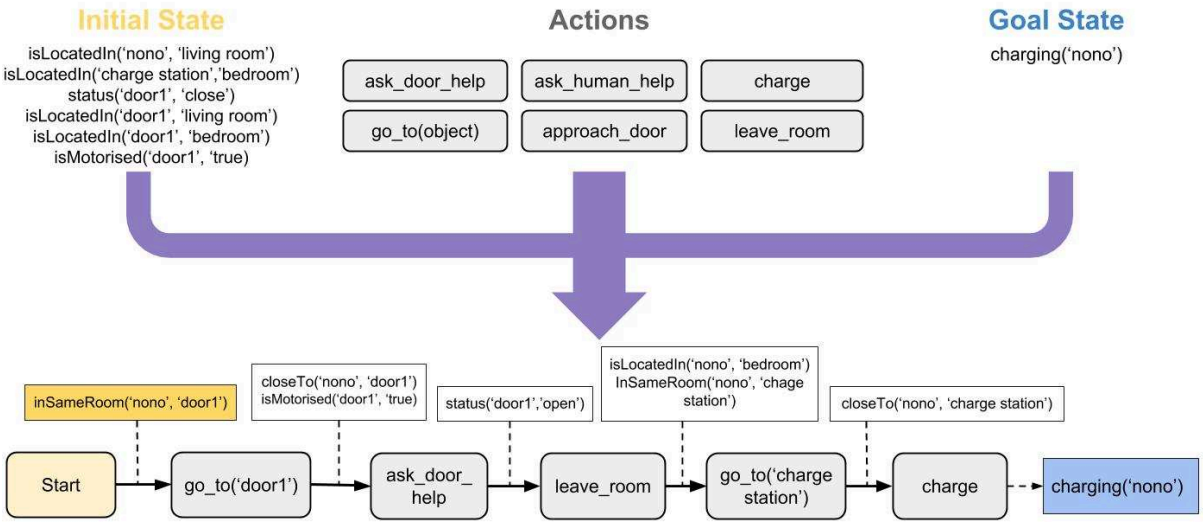


Figure 5.1: Example of a STRIPS planning process

In this scenario, Nono the robot proceeds to recharge its batteries. Hence, it aims to reach the state where it is charging. STRIPS generates a sequence of actions that allows to go from the initial state to the goal state. It uses the available actions (or tasks) and their preconditions and postconditions (not represented). Intermediate states are partially represented.

5.2.2 Hierarchical Planner

A major derivation of STRIPS is Hierarchical Task Network (HTN) [153, 170, 48], or more generally, hierarchical planners. Such planners rely on a richer knowledge than STRIPS-like planners. In fact, HTN considers two types of tasks (or actions): primitive and composite. Primitive task can be executed as such, similarly to a STRIPS action. Composite tasks, on the other hand, are tasks that can be composed in subtasks. Consequently, HTN also carries the knowledge of the possible decompositions, or sub-plan, for composite tasks: by construction, this knowledge is a hierarchical network of tasks. Each sub-plan is associated with preconditions using predicates. HTN algorithm consists in selecting decompositions for composite tasks according to the context until only primitive tasks are remaining. By relying on the knowledge of a task network, HTN requires more specifications from the designer. However, this reduces the search space, making HTN usually quicker than STRIPS. Furthermore, by specifying the decompositions, the designer has more control on the planning and is ensured of the quality the generated plan. For these reasons, HTN and hierarchical planners are popular, particularly in robotics. Similarly to STRIPS, HTN was used and improved in various contexts. It also has numerous implementations [49, 38, 123, 124].

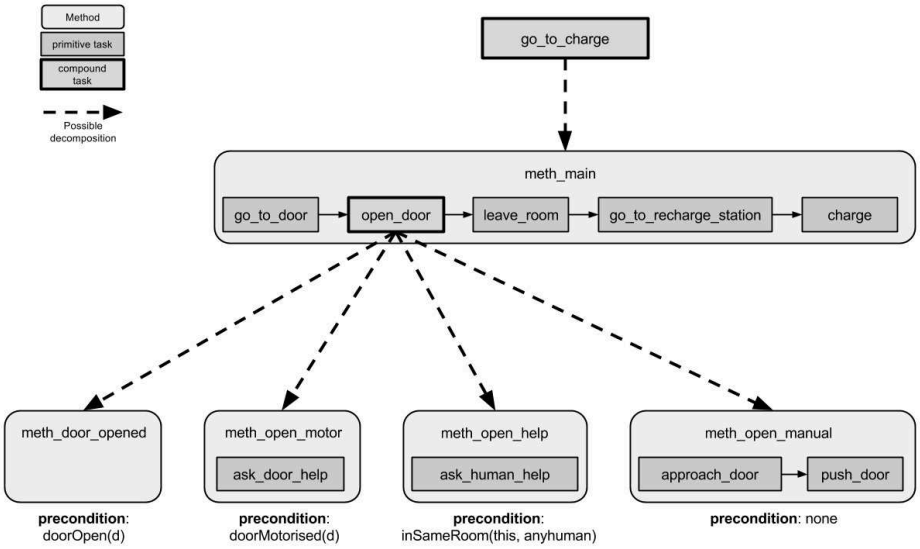


Figure 5.2: Example of a HTN

This knowledge represents the tasks and methods for the robot, Nono, to go to recharge. The composite task go_to_charge has one single method. This method is composed of five tasks. The composite task has five methods that represent different options to perform this task. When planning, HTN will select one of the methods whose preconditions are valid in the current context. Of course, in practice, HTN are much larger, but a full HTN can hardly be displayed in this document.

5.2.3 Probabilistic Planner

Partially Observable Markov Decision Process (POMDP) can also be used as probabilistic planner [83, 14]. POMDP represents states of the environment and probabilistic transitions according to observations. The plan is generated by applying iterative methods such as value iteration and policy

iteration. The main advantage of POMDP is its ability to generate a plan under uncertain and partial observations, making them popular and relevant for service robots. However, they are complex, computationally intractable and require a complete planning knowledge from an expert. Consequently, similar to FSM and ID, they are adapted for relatively simple or specific cases.

5.3 Task Planning for Personal Robots: Problematic

The presented classes of planners are generic and can be easily adapted to applications. In fact, there exist tens of works and improvements over multiple constraints. For instance, performance and efficiency of planning is commonly addressed. Service robotics carries very specific constraints for task planning, particularly in home environments. Performance is often not an issue for robots, in fact, in comparison, task execution is much longer and makes the planning time relatively negligible. In fact, on the one hand, for a robot, performing a physical task, such as opening a door or simply go to another room, is usually a matter of minutes. While on the other hand, most plans are generated in a few seconds.

However, task failure is an issue for service robots. In fact, in home environments, robots may encounter various hazards leading them to fail one of the planned task. A failure can have various causes:

- Breakdown: the robot has a hardware or software issue. It can be a fall[1], a battery depletion, actuator malfunction or others. Nao robots are, for instance, commonly subjected to overheating.
- Context change: due to another actor, such as a person, the context changes making the current plan outdated. For example, the robot may have to grab an object, but a person takes it before the robot could.
- Robot limits: the robot is unable to perform a task as its capabilities are limited. For example, its arms maybe too short to grab an object it has to collect.
- Non-expected failure cause: a task fails due to environment constraint that were not expected when designing the system. For example, vocally warning a deaf user will always fail...

Whenever a failure occurs, the robot shall find an alternative plan to reach its goal. A simple and common approach is to regenerate a new plan with updated context information. Even if replanning is not time-consuming, the robot "wasted" the actions performed until the failure, and has to do more tasks than expected to reach its goal, making the plan execution much longer and resource-consuming. Moreover, replanning is sometimes not enough, and more complex and time-consuming computation maybe required [57], slowing even more the robot. On top of that, as such robots are likely to serve persons, it is worth considering impact on the user acceptance. In fact, if a robot is too slow or keep failing, the user will not be willing to use it. Furthermore, for healthcare applications failing can be unacceptable as some tasks may be critical, medication delivery for instance. For robots operating in smart homes, it is important for task planning to take into consideration the possibilities of failure in order to avoid them and/or handle them properly.

[1] https://www.youtube.com/watch?v=g0TaYhjpOfo

Another important dimension is the context awareness. In fact, unlike many applications, in smart homes the context knowledge is variable and may not be complete. As sensors may not be available or reachable from the robots, it may have an incomplete knowledge when planning. The knowledge may also change during execution. Oppositely, various data are irrelevant in plan generation. For example, the temperature of a room has no impact on the plan generated for delivering medication. Yet, obtaining such a data has a cost in term energy and processing, that is pointless if the obtained data is not used. Thus, for robots in smart environments, it is important to consider the dynamism of the context, as well as the observation needs for the creating the plan.

We will now review state-of-the-art planners adapted for domestic robots and discuss them over the aforementioned constraints.

5.4 Robotic Task Planning for Home Environment

Task planning is essential for service robots. It is consequently a commonly addressed problem. Obviously, contributions go beyond the failure management and context awareness as numerous other key problems are on the table, such as HRI.

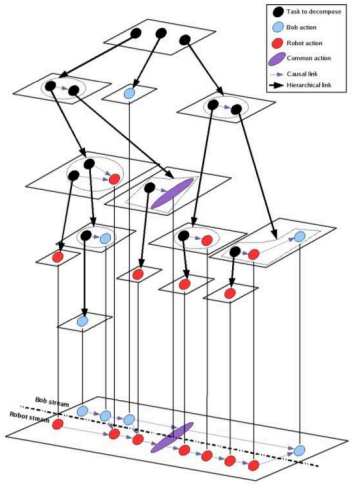


Figure 5.3: HATP Refinement Tree and Time Projection from [118]

HTN was widely used in such a context. In fact, its hierarchical structure offers interesting possibilities of control and understanding. As the matter of fact, Montreuil et al. proposed Human Aware Task Planner (HATP) [118]: HATP is an HTN upgrade based on Simple Hierarchical Ordered Planner 2 (SHOP2) [124] implementation and it aims to enable socially acceptable plans for multiple agents, including robots and humans, but also possibly smart devices. To do so, it relies on social rules that define undesirable states, undesirable sequence of actions, how to balance effort between actors, and other factors. The main difference between HTN and HATP is that HATP does manipulate a refinement tree, on top of the standard task set. By manipulating a tree, as shown in Figure 5.3, HATP is able not only to find a plan, but also to find the best socially acceptable plan. Besides, HATP respects time constraints, essential when interacting with a human user. By using social rules, HATP as an extended social context awareness, allowing it to efficiently generates socially acceptable plans. However, it is to notice that failure management is not explicitly addressed, while failure can be a critical social acceptance criterion.

Later, Lallement et al. created Hierarchical Agent-based Task Planner (HATP) [98]. HATP is also an HTN upgrade that aims to provide an adapted planner for roboticists. It uses the "previous HATP" contributions HATP features are numerous: it provides a convenient object oriented world representation, an intuitive domain formalism, social rules to generate acceptable plan, and the possibility to interleave HATP with a geometric planner. HATP eases the work of the designer thanks to its user-friendly formalism but is also able to generate extremely fine plans through its interface with a geometric planner, a tool that generates plan for low level tasks, such as grasping orientation with

the robot arm. This approach is able to use an extended context, including social and geometric data, thus providing better and finer plans. However, having such a precise and constrained plan make it more sensitive to context changes. Such planners are currently being improved and tested in follow-up works [179].

Consecutively, HTN was also improved to provide human-robot collaborative plans [116] for shared tasks, as depicted in Figure 5.4. In this work, the HTN hierarchical structure is fully taken advantage of, in order to share and explain the plan. In fact, in this approach, sub-plans can be executed by different actors. The planner is able to affect an actor to a sub-plan thanks to the knowledge of the expertise of the actor. It also uses social rules and a cost model proposed in previous works. Furthermore, if a human is a novice for the given task, the robot is able to explain to her/him how to proceed using the known decompositions of the task. Consequently, as actors may be learning how to perform tasks, they are likely to fail. In this work, the robot monitors the user's actions and replans in case of failure, however, it only explains the part that went wrong, making the interaction more acceptable. By monitoring the user, the robot also assesses his/her expertise level and uses it on its planning process.

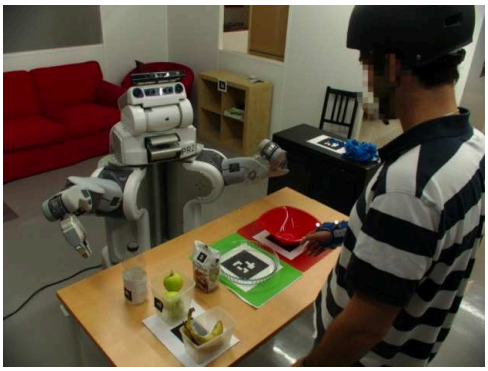


Figure 5.4: Example of human-robot collaboration for preparing a meal, from [116]

Although these approaches underline the possibilities of HTN and the context awareness for robots at home, they do not tackle the problem of failures in plans. Hopefully, multiple works were conducted towards it.

Weser et al. [185] addressed the problem of task planning in partially observable and dynamic environments. Their work is part of the design of the architecture of the TASER robot. Again, this approach relies on HTN and uses SHOP2 [124]. As mentioned in the previous section, the robot may not have access to all the context information, leading to an incomplete knowledge that can make plan generation inaccurate. In order to overcome this, Weser et al. propose to generate a specific plan to acquire missing information, before replanning with the now updated context knowledge. To do so, they define a dedicated replanning action: whenever the plan reaches this action, the plan is regenerated. But as they noticed, "replanning is a resource and time-consuming process". Although this is arguable, we agree that replanning should be prevented as discussed in the previous section. To prevent this drawback, TASER's planner tries to anticipate perception results using assumptions and hypotheses. For instance, if doors are usually closed, a door status will be assumed as closed by default, even if not observed. If the anticipation was correct, the plan does not need to be rebuilt, otherwise, it is handled as usual by replanning. By including perception tasks in the plan, TASER's architecture enables dynamic context-awareness and efficient plan regeneration.

The work conducted by Hanheide et al. [71] aims to provide a global planning solution for robots operating in an open and uncertain environment. They explicitly address the issue of explaining and handling task failures. The approach consists of comparing the actual and expected context observations, and to discover which unexpected data explains the failure of the task. To do so, whenever

the robot encounters a failure, it generates a dedicated plan and uses a diagnostic knowledge in order to extract failure causes. The diagnostic can rely on the assumptions made by the robot through assumptive actions. Figure 5.5 shows an example of dependencies between tasks and assumptions. Afterwards, the robot is able to make a new plan that avoids the identified causes. Actually, this work refines the replanning process thanks to a decent understanding of the failure, yet, it is not stated that this acquired knowledge is used for future plans.

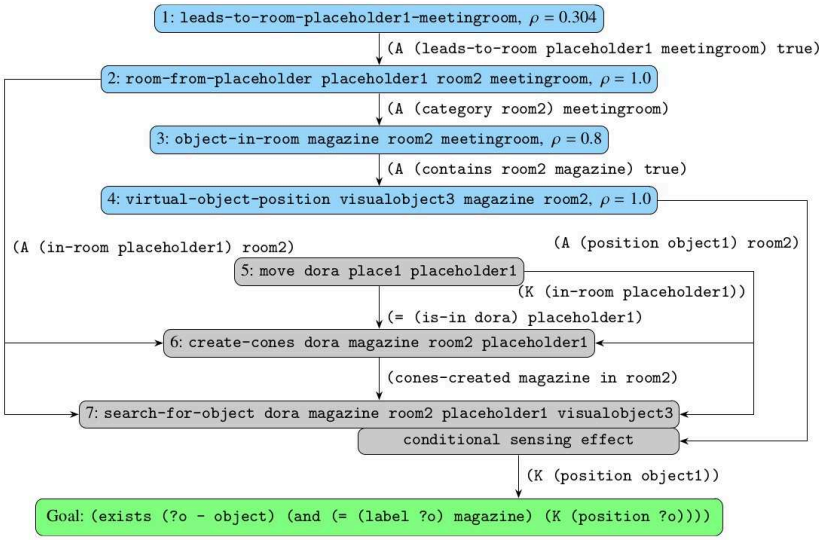


Figure 5.5: Example of action dependencies from [71]

Assumptions, obtained by performing assumptive action, are coloured blue, physical actions are coloured grey. Arrows indicate that the first action has an effect which the second action depends on via preconditions or conditional effects.

Another approach that handles dynamism, is repairSHOP [183]. Although not conducted in a robotic field, this work is relevant and can be applied in such a context. RepairSHOP was conceived on top of the SHOP [123] HTN planner. Since SHOP is not able to deal with context changes aside from replanning, repairSHOP aims to provide on-the-fly plan adaptation. To do so, they created a specific structure: the goal-graph. It maintains the dependencies between HTN task nodes, allowing SHOP to monitor changes in task's conditions. In fact, any context change is propagated through the goal-graph to the related tasks. SHOP is then able to replan from the designated task and only affects a part of the current plan. By relying on an independent structure, repairSHOP needs a mapping between HTN and the goal-graph. Unfortunately, how the context is acquired during the execution is not addressed in this work.

As mentioned earlier, Robot-Era project conducted to the proposition of its own planner [40, 41]. The planner was designed in accordance to the constraints of robots operating in a smart home with various actors, making this work essential for us. In fact, this planner cope with multiple requirements. Hence, it is needed to handle multiple goals, to use high level reasoning mapped to low level actions, to ensure the coordination and collaboration between actors, and finally, to ensure the share of resources and the schedule. Unlike previous works and due of its strong constraints, it was designed from scratch and differs from classical formalisms, from STRIPS or HTN for example. In

this approach, actions are defined as *activities*, that are not related to the user activity we use in our work. An activity is associated with preconditions and postconditions, but also with a time interval where it can occur, and with the resources used by the activity. The planner relies on the notion of *constraint network*. A constraint network is a set of activities associated with time constraints, including "meets", "precedes", "during", and ten other. A *configuration plan* is a plan that is feasible in term of consistency, time and resources. As implied, the generated plan is much more than a simple sequence of actions and affects the whole smart environment. The planning process is actually similar to HTN. In fact, their approach takes as input a goal constraint network and relies on decomposing activities into sub-plan, that is to say sub constraint network. This allows to go from generic high-level plans to a very specific and detailed one, as depicted in Figure 5.6. However, as it has to cope with a lot of constraints, the planning process is complex and relies on five solvers that interact together:

- A **temporal solver** to ensure time consistency.
- A **resource scheduler** that guarantees that no resource is overused.
- A **state variable scheduler** that aims to avoid conflictual context states.
- An **information dependency reasoner** that enforces the information dependencies between functionalities.
- A **causal planner**, that is a more classical planner using preconditions and postconditions.

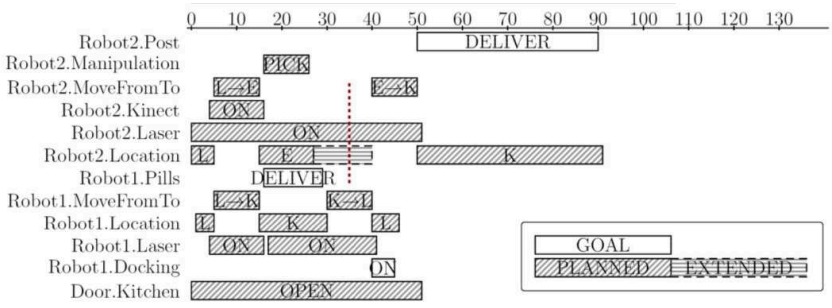


Figure 5.6: Example of plan generated by Robot-Era planner, from [40]

The plan takes into account multiple actors as well as time constraint.

Furthermore, when the plan is being executed, the approach shows to be flexible. In fact, their approach alternates between planning and plan monitoring. The monitoring consists of sensing and updating the configuration plan according the observation. A natural example is a delay in the execution of an action, that implies a change of schedule. The solvers monitor the evolution of the plan, whenever an essential context data changes, the replanning is triggered to adapt the plan. This allows coping with new goal, context changes and possibly failures. Note that, although it adapts the plan continuously, the planner only reacts to changes or failures, and does not prevent them. Robot-Era's planner offers a solution that supports many constraints imposed by the smart home environment. However, such features imply a very complex and specialized solution that uses its own dedicated formalism, making it hard to upgrade and use.

As mentioned, uncertainty matters in home environment, accordingly POMDP were used for robotic planning. Zhang et al. [192] used POMDP associated with logical inference. In this work, Answer Set Prolog (ASP) is used to model the context knowledge. ASP is a declarative language that supports non-monotonic logical reasoning. Although it does not support uncertainty, it enables a rich representation of the context as well as inference, which is then used to generate a multinomial prior for the POMDP state estimation. POMDP on the other hand, supports well the uncertainty, on the contrary to any reasoning capabilities. In this work, hierarchies of POMDP are used in order to adapt the vision and navigation process to the current task. However, in that case, the planners mainly focus on vision and navigation. Yet, the scope of this research is mostly on the context knowledge modelling and management. It proves the usage of reasoning and inference to enhance the robot planning process.

Finally, only a few works try to address the problem of avoiding failures based on past experience. To the best of our knowledge, the works led by Sariel and Kapotoglu [157, 86] are the main ones in this field. In this approach, based on previous failed situations, the robot is able to determine failure causes and avoid using tasks that are expected to fail. An illustrative scenario can be found in Figure 5.7. To do so, the authors use Inductive Logic Programming (ILP), an experiential learning framework that builds an experience by deriving hypothesis from failure situations. Hypotheses can be seen as rules that assert if a task is likely to fail or succeed when observing some context data. Each hypothesis is associated with a probability that is adjusted according to the experience: a low probability meaning there is a lot of ambiguity. Whenever a task ends, context observations are stored and labelled as success or failure, this knowledge can be seen as the experience. The hypotheses and their probabilities are then adjusted accordingly. With these hypotheses, the planning process is adjusted to prevent future failing situations. They use a POMDP planner that was adapted to take into account the hypotheses, thus taking advantage of POMDP probabilistic approach and its natural compliance with hypotheses' probabilities. This technique has proven to improve the efficiency of planning for the object manipulation case study. However, in a more complete home scenario, this approach lacks a suitable context model. Furthermore, it is subjected to bias, in other words, it may identify redundant yet related context data as failure cause or miss some in case of multiple possible failure causes.

All in all, reviewed approaches characteristics and features are summarized in Table 5.1 according to the previously determined criteria: context awareness and failure handling. They are discussed in the following section.

5.5 Conclusion

In this chapter, we reviewed automated planning in regards of our objective for a personal robot interacting with a smart home. After quickly setting aside non-planning based alternative, such as FSM or BT, we classified planning approaches into three classes: chain, hierarchical and probabilistic. We underlined that standard planners would be subjected to failures and suited well for combination with a smart environment. Therefore, we studied planners specifically designed for robots or that provide key features. Their features are summarized in Table 5.1. We notice that most of them rely

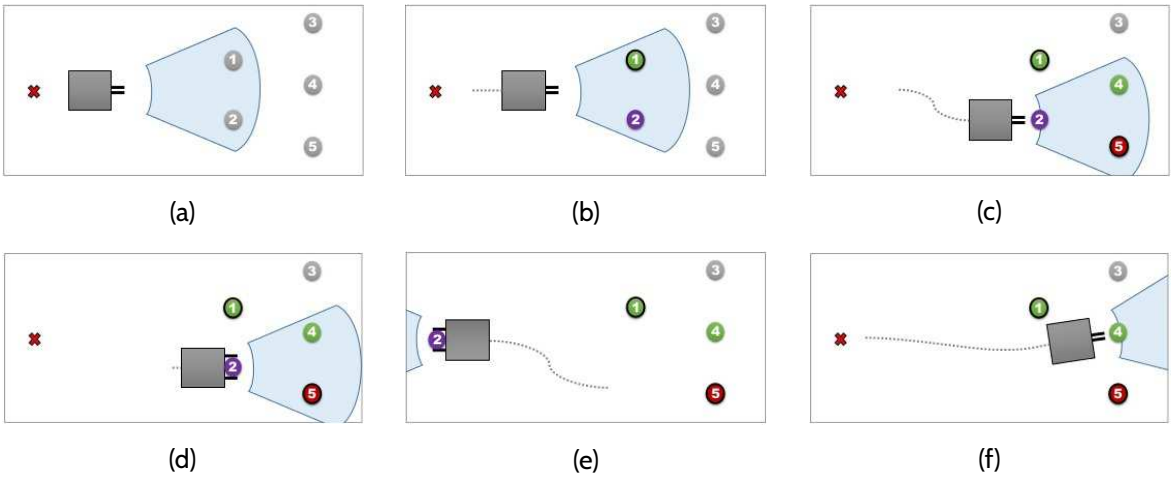


Figure 5.7: Illustrative scenario from [157]

The robot is asked to fetch the object 4, but first has to move obstacles out of the way. Objects coloured grey are unseen. Objects circled with bold edges denote an object with a low probability of *pickUp* task success based on previous experience. Thus, the robot generates a plan that avoids the task *pickUp* on these objects, thus it moves the object 2 and not the object 1 before moving to the object 4.

on HTN or POMDP. In fact, the hierarchical structure enables a controlled plan while probabilistic planner can support uncertainty of observation.

For most planners, when encountering a task failure, the solution is to regenerate the plan. However, reacting to the failure is not enough and we want robots to be able to avoid failures. This implies being able to face the dynamism of the environment and being able to understand causes of failures to prevent new ones. As for dynamism, only two works [185, 183] are explicitly trying to avoid replanning to some extent. About learning the failure causes and enriching the planner with the knowledge of these causes, to the best of our knowledge, only Sariel et al. [157] explored this possibly on a very simple case. Yet, no planning solution is able to cope with failures due to both dynamism and unplanned causes.

Context awareness, although essential for task planning, is not commonly addressed and most techniques use a basic knowledge, sometimes enhanced with specific knowledge to cope with their needs. Some works [192], however, use reasoning and inference to infer high level context data. We also saw that, apart from [40], none of these works addressed the constraints of the smart environments. Thus, taking into account the smart environments and a high-level context knowledge is to be explored.

All in all, although some works propose specific interesting features, none of them is suitable facing our objectives. That is why we proposed two contributions to cope with task failures and use efficiently the context knowledge provided by the smart environment. First, we designed a new-HTN based planner: DHTN. By taking advantage of HTN hierarchical structure, our approach provides a novel planning paradigm that alternate planning and execution. By doing so, DHTN is able to take into account the last changes in the environment and adapt in case of a task failure instead of replanning. On top of that, this novel planner is able to continuously update its context knowledge and compute the subset of the context it opportunistically needs to advance in plan. However, the

	Planner Type	Context Knowledge	Context Dynamism	Failure Handling
HATP [118]	HTN	Standard Social rules	No	No
HATP [98]	HTN	Standard Social rules Geometric data	No	No
[116]	HTN	Standard Agent expertise	Partially	No
TASER's planner [185]	HTN	Standard	Yes	Partially
[71]	STRIPS POMDP	Standard Diagnostic know.	Partially	Partially
repairSHOP [183]	HTN	Standard Goal graph	No	Yes
Robot-Era [41]	Custom	Standard	Partially	No
[192]	POMDP	Enriched	No	Partially
[157]	POMDP	Limited	No	Yes

Table 5.1: Summary of features facing the dynamism of the context and failure management

Note than a partial handling of context dynamism means the approach only tackles particular aspects or context data. Failure management can always be performed to some extent by replanning, yet only approaches explicitly trying to avoid failure and replanning are denoted as managing failures, or denoted as *partially* if limiting replanning.

planning knowledge of DHTN can hardly be exhaustive, and failures may occur due to unplanned conditions. To cope with that issue, we designed a learning-based approach that is able to identify failure causes from experience and causal induction. Our contribution, LEAF models the problem as a multi-armed bandit one and includes users' feedbacks. LEAF allows to enrich the planning knowledge as the robot acquires experience. The planner then avoids to execute task if their failure causes are observed, preventing further failures. These two contributions are presented in Part III.

Chapter 6

Conclusion

In this first part, we analysed the state-of-the-art to refine challenges and identify unsolved problems. We reviewed general usage of service robots in smart environments and pinpointed class of problems. Robots interacting with ambient intelligence are facing challenges of network communications, Human Robot Interaction, localisation and mapping, context awareness and task planning. Our review, including major projects of the domain, shows that context awareness and task planning are two fields that are facing critical and essential challenges. Consequently, we have focused particularly on these two points.

Firstly, we reviewed the literature around context aware techniques used for activity recognition and situation awareness in domestic context. These techniques can be divided into two categories, learning-based and specification-based approaches. We noticed that each has it own advantages and drawbacks facing our requirements. Thus, we explored the possibility of hybrid approaches. And indeed, techniques using both learning-based and specification-based appear to be promising and rely on the strength of both type of approaches. However, none of them was able to completely fulfil all our needs. The proposition of D'Aniello et al. [45] is the closest to cover all our challenges, but they do not over all the uncertainty issues.

Secondly, we analysed task planning techniques. Widely used in many applications, the standard approaches are facing major limitations as for failures and context awareness in our case, independently from the type, namely, chain, hierarchical or probabilistic planner. In consequence, we focused on a planner specifically designed for personal robots. We noticed some very interesting propositions, but each of them was isolated. For example, Weser et al. [185] planner allows to observe the context at runtime, but relied on replanning, and oppositely Kapotoglu et al. [86] approach prevent failures but does not address context acquisition. In the end there was no planner that avoids task failures and takes into consideration the context acquisition.

From these observations, we proposed five contributions to overcome these omissions. Our works cover the whole spectrum of challenges, tackling perception, cognition and action. This results in a modular framework for personal robot in smart environment: FAIRIE. FAIRIE general process can be seen in Figure 6.1. Each of our contributions is actually a module usable by FAIRIE, as well as other ones from other researchers. These contributions can be divided into two classes that matches the domain we reviewed. In the first following part, we address our perception and cognition contributions for context acquisition and decision making. Then, in the second part, we focus

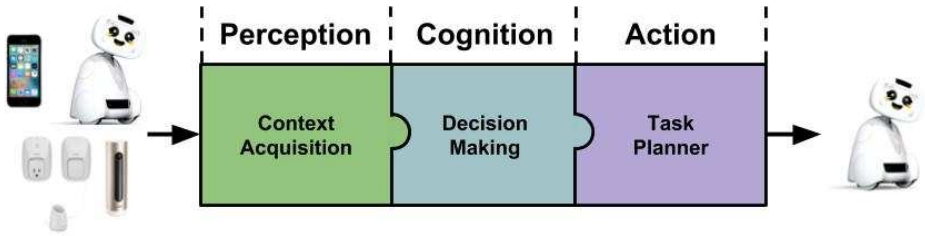


Figure 6.1: General process of FAIRIE

on our proposals toward failure avoidance in actions and task planning adapted for robots in smart environments.

Part II

Context Perception and Cognition

Chapter 7

Introduction

Being aware of the context is an essential matters for domestic robots. The knowledge of the context enables the robot to understand its surroundings, to take a decision and to act. Thus, the first step of any intelligent system, including FAIRIE, is to perceive and understand the context. By using the smart environment, personal robots have access to diverse information about the environment, in other words, they can acquire a rich context knowledge. However, such an approach also brings problems. As the devices are various and numerous, the issues of heterogeneity, flexibility and reliability arise. In fact, the context acquisition should be compatible with heterogeneous data, for example supporting events from motion sensors as well as recognized gestures from the robot cameras. As the smart environment is open, context acquisition should not be restrained to specific devices. But the most problematic issue remains the uncertainty of data. Context data can be imprecise, inaccurate, outdated, in contradiction or missing. This alters the understanding and decision process of the robot. While there exist techniques that cope with heterogeneity and flexibility, approaches that additionally tackle uncertainty dimensions are less common. Actually, as we discussed in our analysis of the literature, there are no solutions that cope with all uncertainty dimensions.

That is why we proposed several contributions to enable context awareness over uncertain and heterogeneous context data. In this part, we describe the perception and cognition phase of FAIRIE, that allow to go from sensory data to a decision. We first address context acquisition through two contributions. Context data can be provided in different ways, in particular through events. We consequently proposed a specification method that batches, semantizes and fuzzifies events to generate a high level event that cope and/or model four uncertainty dimensions. We also considered context data acquired from videos. We particularly focused on the case of a user activity identification from the robot cameras. Combined with other context data provided by smart devices, the vision-based recognition process was made more accurate and precise. Secondly, we describe our cognition contributions. We proposed an approach, based on a hybrid technique, that analyses the acquired context knowledge to identify anomalous situations and help the robot takes a decision accordingly. Through these contributions we cope with a aforementioned challenges, as we will see in the two next chapters.

Chapter 8

From Sensor to Knowledge

8.1 Introduction

Knowledge is the foundation of any intelligence. For service robots and ambient intelligence, understanding the environment is a key for suitable behaviour and reaction. Consequently, acquiring the knowledge of the context is the first, and possibly most critical, step. Without a proper knowledge robots and ambient intelligence, similarly humans, are prone to mistakes. The context is mainly observed through sensors, that provide a precise information about the environment, namely context data. For example, Nono can use its cameras as well as motion sensors to observe the environment. By gathering these context data from various sources, robots and smart environments perceive the state of the surroundings, and create a knowledge, from the sensors. This knowledge can then be used to infer new context data, used in the cognition phase to perform decision making, or used in the action phase to generate a task plan.

However, as illustrated in our scenario of Arthur and its robot Nono, perceiving the context is not a trivial task and multiple constraints arise. Firstly, it is important to consider that provided data are not perfect, for example, Nono can observe Arthur in a room, while a smart device states the opposite. Similarly, Nono may be mistaken in its observation. In fact, the quality of data is not ensured, thus uncertainty is an essential matter. As defined in Section 1.2.2, we consider five dimensions of uncertainty [190]: freshness, accuracy, precision, completeness and contradiction. Secondly, it is worth noticing that the smart environment is open and filled with various sensors. Typically, Arthur home is equipped with cameras, motion sensors and thermometers, and he may add new devices such as beacons. Hence, when observing the context, it is essential not to make the assumption of having specific devices, and to equally gather all types of data, that can be event based or video stream for example. Lastly, some context data may be complex, such as the user activity. For instance, when Arthur is cooking, Nono needs to rely on thermometers, microphones and its own cameras to infer Arthur's activity. Thus, acquiring the context also consists of managing inferred and possibly complex context data.

The literature provides numerous techniques for context acquisition and management, yet, tackling the aforementioned challenges is an open problem. Context acquisition is usually associated to a particular purpose, mostly recognition of the activity of the user or obtaining particular context

data, but can be applied in more general cases. Techniques can be divided in two categories [190]: (1) learning based or data driven approaches, that rely on machine learning and low-level data, and (2) specification based or knowledge based approach, that use the provided knowledge and operate on high level data. On the one hand, learning based approaches appear to support well uncertainty, as some techniques, such as CRF [97] or DT [154], can support up to three uncertainty dimensions. However, they are mostly not compatible with the open aspect of the environment due to their stasis and only use low level context data, such as raw events from a particular motion sensor. On the other hand, specification based techniques, including formal or spatio-temporal logic [74, 11] and ontologies, can model and cope with complex context data, such as the user location or activity, and can be adapted by creating or adjusting rules. However, uncertainty management is limited in those approaches. To overcome the limits of both approaches, hybrid techniques were proposed [39, 151, 45, 10, 128] to specifically manage the context in smart environments. Although these approaches perform better, in particular, by tackling more uncertainty dimensions and high-level context data, they still can't cope with all our challenges.

In this chapter, we present our solutions for acquiring the context knowledge for a robot in a smart environment. As we will discuss in the next section, we do not tackle communication and interaction with devices; we actually focus on the management of the data required to provide the context knowledge under the constraints previously illustrated. Consequently, we proposed two solutions. Firstly, we propose an event-based context acquisition approach that semantically enriches and fuzzyfies events to model and tackle four uncertainty dimensions. Our approach actually transforms raw events into complex, fuzzy, and semantic events that can then be stored and used, enabling more accurate inference of context data, in particular user activities, and an improved cognition phase. Secondly, as not all sensors are event oriented, we also proposed a refined vision-based activity recognition process. Indeed, vision is the main source of information of the robot and can be used to recognize the gesture of the user, yet, this solution is highly inaccurate. Consequently, we refined it by using the data provided by other sensors in the environment. These two approaches allow to acquire the context from event-based sensors and robot's cameras, covering most of the spectrum of sensors.

Before reviewing our two contributions, we will first discuss preliminary notions and definitions that are required and used in both solutions. We end this chapter by presenting the experiments conducted and the obtained results.

8.2 Preliminaries

Before addressing our two contributions, it is important to clarify our hypotheses and to define notions that will be used afterwards.

8.2.1 Data Sources

For a robot in smart environments, sources of data are diverse and numerous. To observe, the robot can not only rely on its sensors, but also on smart devices dispatched in the environment. This variety

induces heterogeneity of devices, protocol and data. For example, a motion sensor provides events when a movement is detected, a thermometer provides a temperature on demand, and a camera can perform heavy processing to identify a user. There are two different classes of data sources:

- *Active data source (push mode)*: an active data source or active sensor **sends** (pushes) the information to the system. This is typically the case for event oriented sensors, such as motion detector.

- *Passive data source (pull mode)*: a passive data source or passive sensor **is asked** to provide a data. In other words, the data is pulled. This is typically the case for sensors that continuously analyse the context, such as thermometers.

In a smart environment, it is likely to find both passive and active sensors. Furthermore, communication protocols are also various, and often proprietary. Nevertheless, all data sources should be used for context acquisition.

Although integration and configuration of devices in smart environments is an open challenge [40], we do not address this issue. In order to gather the data, we actually rely on middleware or home-made bootstraps that push data to our solutions. If a sensor is passive, its behaviour is abstracted as an active one. For example, middleware can pull a temperature, and push it periodically or when the value changes. Consequently, we suppose data to be pushed as input in our solutions. By doing so, we put aside the heterogeneity of sensors and protocols, but the heterogeneity of data remains: this led to the usage of ontologies, that can model numerous types of data, as we will discuss in the upcoming sections. The pushed data can then be transformed and handled as events, enabling the usage of Complex Event Processing techniques, as we will see in the next section.

However, not all data can be pushed and transformed as events. This is particularly the case of cameras, in particular the one embedded on robots. In fact, cameras are key sensors for robots allowing them to observe the environment wherever they are. However, as such, cameras only provide pictures or video streams without carrying information, thus, vision algorithms need to be applied to extract data from cameras. As a result, we consider two types of data: events obtained from pushed data, and video streams provided by the robot cameras or others. This led to two separate contributions to acquire the context: one oriented on events, the other centred on the robot vision.

8.2.2 Background Ontologies

Ontologies are commonly used tools for maintaining knowledge. They can model numerous type of data, in particular through the RDF format, that is based on triples, subject, predicate, object. On top of that, they are often associated with inference engines enabling reasoning. For these reasons, ontologies offer a suitable format for carrying the context of the knowledge. But ontologies can also be used to store general information used by our solutions, in that case, they are refereed to as **background ontologies**.

Background ontologies aim to carry structure and information required by our tools. The structure, meaning the classes and their properties, allows to normalize our data, allowing an easy integration into a context ontology. On the other hand, background ontologies carry other information

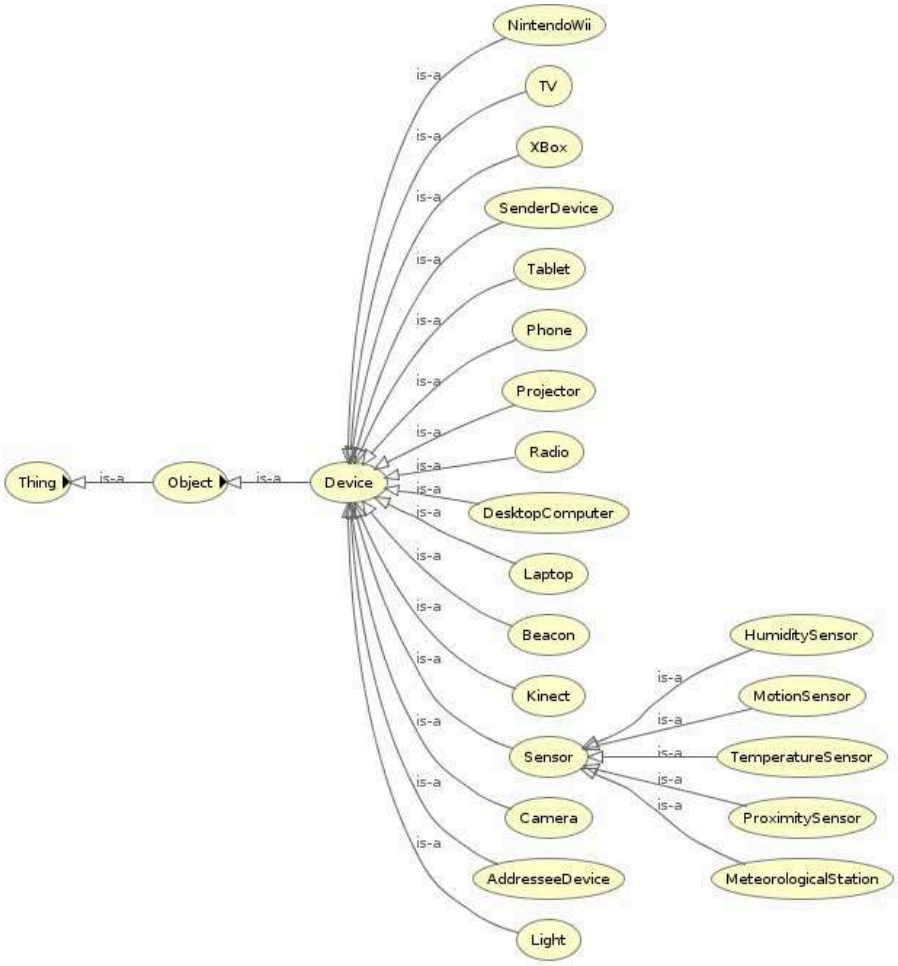


Figure 8.1: Excerpt of a possible background ontology

This excerpt shows the types of devices in the ontologies that are used to classify objects in the environments.

about devices, robot, environment and others. Typically, the background ontology can carry the location of sensors, the reliability of class of sensors, or specific sensors, or even the topology of the environment. An example of carried background information can be found in Figure 8.2. Such an ontology must a minima carry the following features:

- Class *Sensor*, and subclasses of *Sensor*, such as *MotionSensor* or *thermometer*
- Data property *hasConfidence* associated to *Sensor* subclasses that expresses the reliability degree for a sensor. This trust is supposed to be provided by an expert. It can be learnt statistically, but this remains out of the focus of our work.
- Data property *hadId* associated to *Sensor*, that links each instance of sensor with an identifier.
- Class *Activity*, that represents the user activity.
- A set of inference rules, provided by an expert and depending on the environment and the ontology. These rules are used to infer new context data, but also to refine the activity recognition, as we will discuss later.

Further concepts and properties can be provided according to specific needs and constraints of one environment; this remains the responsibility of the designer. In our work, we improved a context ontology for human activity representation [1] [151] with the aforementioned classes and properties, as well as new type of sensors and activities. An excerpt of this ontology can be found in Figure 8.1. Note that any ontology that carries the minimal features aforementioned can be used. We used this background ontology in both of our contributions as we will discuss now.

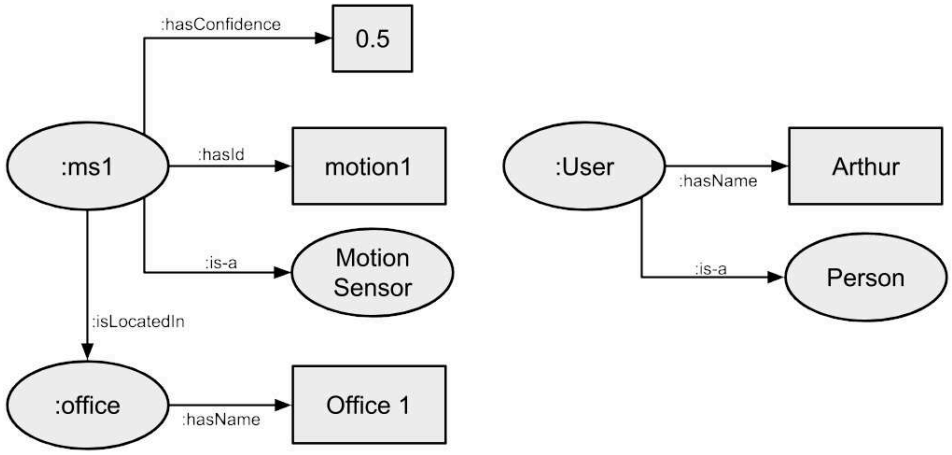


Figure 8.2: Example of two RDF graphs carrying background information

In this example, an expert has provided information about a motion sensor and the user, Arthur. These graphs are part of a background ontology.

8.2.3 Complex Event Processing

Complex Event Processing (CEP) [107] is widely used in research and industry to manage events. CEP techniques consist of generating complex events from other events, simple or complex, based on rules, thus being a specification-based approach. CEP allows to gather, filter and batch events, as well as perform a first reasoning by generating more and more complex events. In our contributions we relied on a CEP as a first step to handle entering events. But what is an event? An event stream? Let us define those notions.

Firstly, the entering and resulting flows of events can be considered in different ways. CEP usually assume an event cloud or an event steam as input. They are defined as follows.

Definition 5. *Event cloud: An event cloud is a partially ordered set of events where dependencies between events are set by criteria other than time.*

Definition 6. *Event stream: An event stream is an ordered time sequence of events.*

Event streams are mostly used, but event clouds can be required in some context. They define how the events enters a CEP and how they are handled within it.

In CEP, there are two types of events:

[1] http://users.abo.fi/ndiaz/public/FuzzyHumanBehaviourOntology/FuzzyHumanBehaviourV11.owl

- Simple event: A simple event is provided by a source external to the CEP. It is a raw event provided by a sensor, and it carries a raw data.
- Complex event: A complex event results from the abstraction and/or aggregation of other several events, simple or complex.

Formally, an event is defined as follows:

Definition 7. ***Event:***
An event is a vector defined as $e = \{t, ids, type, v\}$ where t is the timestamp, ids is a set of sources identifier, $type$ is the source type, v is a value.

- *Simple event: $|ids| = 1$ and t is the time the simple event enters the CEP.*
- *Complex event: ids is the union of all sources identifiers carried by events used to generated the complex event e. t is the time the complex event was generated.*

For example a motion sensor may provide the following simple events $e_1 = \{t_1, \{motion1\}, motion, active\}$ meaning that the motion sensor was triggered and generated an event at date t_1. Similarly, a camera may provide the event $e_2 = \{t_2, \{camera1\}, identity, arthur\}$ if it identifies Arthur. Note that, in our case, we suppose these event formatted by a middleware or an interface. Identifiers are set according to the background knowledge information on sensors. Timestamps are set to the time they are sent to the CEP. Such events can be aggregated to form a complex event. For example, the user location complex event, $e_3 = \{t_3, \{motion1, camera1\}, location, kitchen\}$ can be generated from e_1 and e_2 as the expert knows $motion1$ and $camera1$ to be located in the kitchen. Note that a complex event is not necessarily composed from multiple simple events, in fact, a complex event can simply be a refinement or a filtering of an event. In CEP, complex events are often computed through rules. Rules are provided by an expert and allow to gather, filter and batch events. For example, events of type $motion$ and $identity$ can be gathered, or only events from given sources. As output, CEP rules create event batches.

Definition 8. ***Event Batch:***
An event batch is a group of events resulting from a rule. Batch can be defined according to the number of events, or according to time.

Note that a batch can be an event stream or an event cloud. A batch is defined by time, meaning it carries events that are in a same time window. It can also be defined by the number of events, in that case, the batch is provided when it reaches a given number of events. For example, a rule may generate batch of location events such as:

$$\{\{t_3, \{motion1, camera\}, location, kitchen\}, \{t_4, \{motion3\}, location, office\}\}$$

CEP can be used to gather, filter and batch events, however, most CEP techniques do not tackle uncertainty. In the literature, CEP rarely cope with uncertainty. In fact, the research community is mainly focused on performance [191, 187, 29, 180], as pointed out by Cugola et al. [37]: the few works that do only address a subset of uncertainty dimension. Among these works, Artikis et al. [9] aims to

identify uncertainty dimensions that may appear in CEP and discussed of possible solutions to extend traditional events. However, as their context is different from ours, the identified uncertainty dimensions differ from ours, except imprecision. Wasserkrug et al. [184] proposed a framework for knowledge representation in CEP supporting uncertainty. By relying on probabilities and generated Bayesian Network, they address two types of uncertainty: the imprecision of events and the 'uncertain relations between events'. Other dimensions, that are essential in our context, are not dealt with. Another approach dealing with uncertainty is CEP2U (Complex Event Processing Under Uncertainty) [37]. It combines CEP and Bayesian Network. However, it only deals with, yet greatly, two dimensions: the accuracy of data and accuracy of rules. More recently, SCARG (Sequence Clustering-based Automated Rule Generation) [102] aims to tackle the problem of dynamic changes in the domain and/or expert mistakes. The approach relies on learning, clustering and Markov Probability Transition Models, yet, it does not explicitly cope with uncertainty dimensions. Hence, no CEP techniques can efficiently help us toward the management of uncertainty.

To cope with these limitations, in our solution we used CEP alongside with ontologies and fuzzy logic. As performed by some semantic CEP such as ETALIS [7] of SCEPter [193], we used a background ontology that carries key information to handle uncertainty.

8.3 Context Acquisition over Event Based Sources: FSCEP

For a robot operating in a smart home, possibilities of acquiring the knowledge of the context are as various are the variety of sensors. Most of these sensors, such as motion detectors, beacons or thermometers, provide data that can be abstracted as pushed events. However, in real life, these devices are not perfect, causing the data they provide to be uncertain. In order to acquire the context over uncertain events, we propose a new context acquisition system: Fuzzy Semantic Complex Event Processing (FSCEP). FSCEP, aims to cope and/or model four uncertainty dimensions, namely, freshness, inaccuracy, imprecision and contradiction, by providing high level, semantic and fuzzy events. This is enabled by relying on CEP, background ontologies and fuzzy logic. FSCEP is axed over three main steps, as seen in Figure 8.3 :

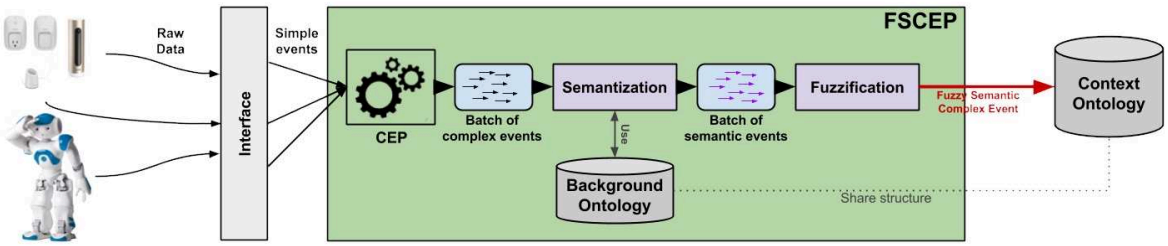


Figure 8.3: FSCEP architecture

1. **Events Acquisition**: Once sensors data are transformed into events and pushed to FSCEP, they are firstly acquired through a CEP. This CEP operates a filtering and a batching of events over time. Thus, this phase provides a batch of fresh and filtered complex events.

2. **Semantization**:The semantization phase transforms and enriches complex events into context data and semantic events by using a semantic background knowledge. Thanks to the usage of a background knowledge, events are formatted, enriched and associated with a confidence value. As a result, it outputs a batch of semantic events.

3. **Fuzzification**: The final phase consists of analysing the batch of semantic event to generate a single fuzzy semantic complex event. This event carries one context data with fuzzy values. This event is the output of FSCEP and can be used or stored afterwards.

Let us now review each step more specifically.

8.3.1 Events Acquisition

As mentioned earlier, we suppose that data are pushed as events to FSCEP. Those simple events are formatted as events whose timestamp t is set as the date of transformation and emission. Thus, FSCEP takes as input an event stream.

The aim of the events acquisition is to handle the input event stream on three axis. Firstly, it aims to gather similar events together. For instance, events related to the user location, such as motion detections, will be considered together, while events related to temperature, will be dealt with separately. Secondly, it should filter **inaccurate** events. In fact, as sensors are imperfect, some events may be obviously wrong and can be filtered. Lastly, it batches events into time-window. This ensures the **freshness** of events as old ones are not considered.

In order to achieve this, simple events are gathered through a CEP. In our work, we relied on Esper [50], yet, it can be substituted by another CEP solution. The CEP relies on rules provided by an expert, that define how to gather, filter and batch events. Let us consider the following rule:

`Select events of` $type$`="motion" and` $type$`="identity" in time_batch(1min)`

This rule simply gathers events of type $motion$ and $identity$ and put them in a one minute batch. The resulting batch will contains events of those types that were provided in the last minute. By defining such batches, only **fresh** events are gathered. Oppositely, if batches were defined according to a number of events, old outdated ones could have been selected. On top of that, in case of overflow, the recent-first policy is applied in FSCEP. Meaning that if the CEP is saturated, old events will be replaced by new ones. Again, this prevents the system to take into consideration too old events. Rules also allow to filter events, let us now consider the following one:

`Select events of` $type$`="temperature" in time_batch(1min) where` $value < 100$

This rule gathers temperature events in a one minute batch, and filter them according the temperature value. In fact, if the temperature is beyond 100°C, it is considered to be an artefact, and should not be considered. By doing such filtering, the rule perform a first check of **accuracy**. Note that this is just a simple example, more complex conditions can be set, for example, it is possible to verify the uniqueness of an event or the relation between multiple ones.

These rules are defined by an expert that has the knowledge of the environment and its characteristics. Naturally, multiple rules can be defined in the CEP. Furthermore, new rules can be easily

added without altering the other ones or the knowledge. Note that, in FSCEP, the CEP are not used to create aggregated complex events, as this is the role of the following phases. In the end, the CEP allows to gather event of same types, filter inaccuracy, and batch into time-window. For each rule, a batch of complex events is created. Those batches are then transmitted to the semantization phase.

8.3.2 Semantization

As its name implies, the semantization phase aims to add semantic to events. The principle is to transform, by using a background ontology, the raw data carried by events into context data, and model them into an ontological format. Consequently, the semantization process consists of transforming complex events carried in a batch, to semantic events, resulting in a new batch of semantic events. It is defined as follows:

Definition 9. *Event semantization process*
Let B_{ce} be a batch of complex events. Let B_{se} be a batch of semantic events. The event semantization process consists of transforming an event $e_i \in B_{ce}$ to a semantic event se_i.

$$B_{ce} \longrightarrow B_{se}$$
$$EventSemantization(\{e_1, e_2, \ldots, e_n\}) \longrightarrow \{se_1, se_2, \ldots, se_n\}$$

where $0 < i <= n$ and n is the number of events in the batch B_{ce}.

The semantization process relies on three core features: (1) extracting and formatting the context data, (2) enriching with background information, and (3) setting confidence weights. Let us discuss those features.

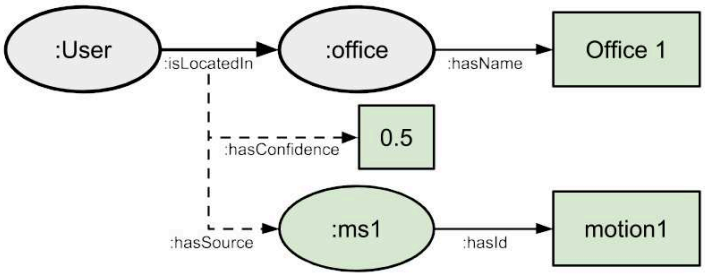


Figure 8.4: Example of a semantic event

Example of a semantic event that carries the user location context data, in grey, as well as related information provided by the background ontology, in green. The confidence value is also attached to the context data as annotation.

The first feature of the semantization is the formatting of data carried by events to context data modelled as RDF . The principle is to create a RDF triple from the information carried by an event. In fact, we define a context data follows:

Definition 10. *Context data (CD):*
A context data is a piece of information about the environment provided by robots' sensors, smart devices

or a knowledge base. Formally, it is a RDF-Triple (subject, predicate, object). $subject$ *is the URI* [2] *of an entity of the environment,* $object$ *is the URI of another entity or a value,* $predicate$[3] *is the relation between the* $subject$ *and the* $object$*.*

For example, *(:user, isLocatedIn, :office)* is a context data that represents the user location. Note that the $subject$ can represent any entity of the environments, e.g. robot, user, furniture or other. The $object$ can represent an entity as well, but can also be a value, such as a number, a name or a Celsius degree. The first step of the semantization is to create such a context data from a simple event. The structure of context are carried in the background ontology. For example, that e location is a relation between an entity and a room as *(?entity, isLocatedIn, ?room)*, consequently, generated location context data shall match this structure. As the background ontology shares its structure with the context ontology, as seen in Figure 8.3, the provided context data match the formalism of the context knowledge and can be immediately integrated. In order to identify what structure to use, matching rules are used between the type of event and predicates. For example, event of type *location* will be associated to the *isLocatedIn* relation. Once the structure is selected, objects and subjects are identified to create the triple. This can be performed by using the identifier carried by event or by relying on hypotheses. For example, we can assume motion sensors only detect the user, that is alone in the environment, thus, thanks to this hypothesis, the subject of the triple can be set as the user. In the end, the simple event is transformed to a context data in accordance to the format of the ontologies. However, the information carried by the event may not be enough to transform it to context data.

Indeed, the imprecision of the event may block the transformation into a context data. For example, as is, a motion sensor does not provide the location of the user, but simply detects a movement. However, the background ontology can tackle this flaw. In fact, the information carried in the ontology can be used to enrich the event data to finally create the context data. Typically, in the case of the user location, the ontology stores the location from all sensors, thus, the user location is inferred according to the identifier of the event. For example, an event $e_1 = \{t_1, \{motion1\}, motion, active\}$ with the ontology carrying the information *(:motion1, isLocatedIn, :kitchen)*, the context data *(:user, isLocatedIn, :kitchen* will be created. In short, information carried by the event and the background ontologies are interlaced to infer the related context data. By enriching data carried by events, FSCEP copes with the **imprecision** issue. On top of that, further information from the background ontologies is associated to the context data. These are also formed as RDF triples and, alongside with the context data, form a RDF graph. Such information can be various and include information about the origin sensors, the type of data, the subject or the object of the context data. Hence, this step leads to the creation of a semantic event, defined as follows:

Definition 11. ***Semantic Event****:*
A semantic event is a couple defined as $se = \{t, G\}$ *where* t *is the timestamp, and* G *is a set of RDF-Triples forming a graph.* G *carries a context data* cd *and information including the sources, identifiers, confidence.*

[2] `https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`

[3] From a RDF perspective, the $predicate$ is an *object property* if $object$ refers to an entity, and a *data property* if $object$ refers to a value.

Hence, a semantic event is a context data extended with extra information, forming a RDF graph. This graph can be used in the fuzzification phase and can be directly integrated in a context ontology. An example of a semantic event can be found in Figure 8.4. Note that this extra information include a confidence value. This value is key in FSCEP, as we discuss now.

The background ontology associates to sensors a confidence value. As implied, this value represents the reliability of a sensors. In other words, it quantifies the accuracy of a data source. Typically, motion sensors can be easily mistaken, thus have a low confidence value. On the other hand, beacons, that detect the user's phone may be more accurate and can be more easily relied on, but only if the user usually stands by his/her phone. The confidence values are supposed provided. In our work, we considered an expert; statistical learning can be applied, but remains out of the scope of our contribution. This value can be set for a class of sensors, such as motion sensors or beacons, or for a specific sensor. When an event is semantized, the generated context data is associated, by annotation, to the confidence value of the source sensor. This can be seen in Figure 8.4. If an event have multiple sources, for example $e_3 = \{t_3, \{motion1, camera1\}, location, kitchen\}$, the average of all confidence values is kept. This confidence value is then used during the fuzzyfication process. By using such a weight, we aim to model the **inaccuracy** of context data.

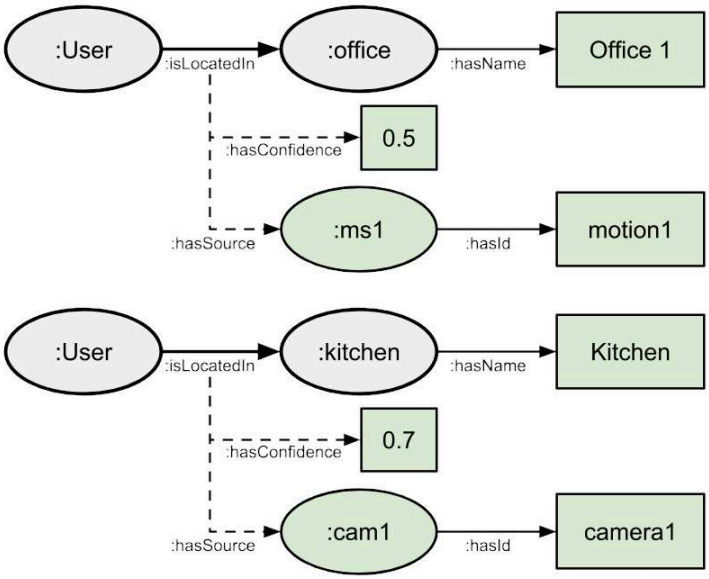


Figure 8.5: Example of a semantic event batch

Note that the two semantic events appear to be carrying context data that are in contradiction: is the user in the kitchen or in the office?

In the end, the semantization process allows to transform complex events to context data and enrich them by using the background ontology. The resulting semantic events are precise and model accuracy through confidence values. They form a batch of semantic events, as seen in Figure 8.5. Note that, as events are grouped per type in the events acquisition phase, one batch carries semantic events representing one type of context data, for example location. This batch is then transmitted to the fuzzyfication phase, that will analyse the whole batch to generate a single fuzzy event, allowing to model contradiction.

8.3.3 Fuzzyfication

The fuzzification process aims to analyse the batch of semantic events to generate one fuzzy semantic event. This event carries multiple values for one context data, each associated with truth weight. By doing so, the resulting fuzzy event models uncertainty, in particular the contradiction and accuracy. The process consists of regrouping semantic events carried in a batch into a single semantic event, and then to compute truth weight based on the information carried by these events, in particular confidence weights. Formally, the process is defined a follows:

Definition 12. ***Fuzzification process***
Let SE_{cd} be a batch of semantic events $SE_{cd} = \{se_1, se_2, \ldots, se_n\}$ where $\forall se \in SE_{cd}$, se carries the context data of type cd. The fuzzification process can be seen as a function that takes as input a batch of semantic events and generates a $FSCE_{cd}$ for a context data type cd.

$$Fuzzification(SE_{cd}) \rightarrow FSCE_{cd}$$

The fuzzification aggregates semantic events into one semantic event that carries a fuzzy context data. Thus, the first step is to compute this fuzzy context data according to the "normal" context data carried in events in SE_{cd}. A fuzzy context data is defined as follows:

Definition 13. ***Fuzzy Context Data (FCD)****:*
A Fuzzy Context Data is a context data (see Definition 10), thus a RDF-Triple (subject, predicate, object) where $object$ is multivalued and weighted. Hence, $object$ is a set of pairs (v, w_v) where $v \in V$, V being the set of all possible values or objects, and w_v is the truth weight associated to v.

For example, considering the batch in Figure 8.5, the following fuzzy context data can be assembled: `(user, isLocatedIn, {kitchen` w_1`;office` w_2`})`. Note that, when represented graphically, each possible value is associated to the subject with its own triple. Thus, a fuzzy context data is actually a small RDF-graph with a single subject, as seen in Figure 8.6. Nevertheless, as for textual representation, we will keep the formalism used in the previous example. To generate such a fuzzy context data, context data carried by events in SE_{cd} are regrouped. As they are part of the same batch, they share the same structure, that is to say the same subject and predicate, but have different objects. The set of these objects, that are the possible values, are detonated as V. The resulting fuzzy context data has the subject and predicate of the entering context data, and as object, it has V associated with truth weights. How these truth weights are computed is the core of the fuzzification phase.

In order to compute the truth weights associated to values, FSCEP applies a membership function. This function uses as input a batch of semantic events and a value, and provides a truth weight for this value. It is defined as follows:

Definition 14. ***Membership Function MF****:*

$$MF(SE_{cd}, val) \Rightarrow WT_{val}$$

where SE_{cd} is a batch of semantic events, $val \in V$ is a possible value for cd and WT_{val} is the truth weight determined for val.

Multiple membership can be applied and FSCEP is compatible with any function that matches our definition. In our work, we considered a weighted mean function based on the confidence values carried by semantic events. By doing so, we aim to compute truth weight that are based on the accuracy of context data. The more accurate the data is, the higher its truth weight is, relatively to other context data. Our membership function is defined as follows:

Definition 15. ***Confidence-based Membership Function*** MF_c*:*
Let A *be the set of all confidence weights* α *of a semantic event* se *in* SE_{cd}*. Let* A_{val} *be the set of all confidence weights* α *of semantic event* se *in* SE_{cd} *where* $object = val$*.*

$$MF_c(SE_{cd}, val) = (\sum_{\alpha \in A_{val}} \alpha)/(\sum_{\alpha \in A} \alpha)$$

Applying this membership function allows to compute truth weight of the fuzzy context data. Let us apply it on the batch of semantic events depicted in Figure 8.5. This batch has two semantic events carrying the context data representing the user to be in the office and the kitchen, with respectively confidence 0.5 and 0.7. Therefore:

$$WT_{kitchen} = MF_c(SE_{isLocatedIn}, kitchen) = 0.7/(0.5 + 0.7) = 0.58$$
$$WT_{office} = MF_c(SE_{isLocatedIn}, office) = 0.5/(0.5 + 0.7) = 0.42$$

This computed truth weight can then be associated to their respective values in the fuzzy context data. In our example: `(user, isLocatedIn, {kitchen` 0.58`;office` 0.42`})`. This truth weight means that the user is considered to be both in the kitchen and office. Trust weights are indeed different as probabilities: on the one hand, probabilities would consider exclusive, meaning the value is actually one or the other, on the other hand, fuzziness represented by truth weights means that the two values are partially true. In fact, the fuzzy context data means that, according to the current knowledge, all possible outcomes can be considered true to a certain degree. This also means that no value should be excluded. Hence, by generating fuzzy context data, we model **contradiction**, but also **accuracy**, as truth weights are based on confidence values.

As entering events are semantic, the formalism is obviously kept during the fuzzification phase. Once the fuzzy context data is computed, a fuzzy semantic complex event can be created. This event semantically represents a fuzzy context data as a RDF graph. Such event is defined as follows:

Definition 16. ***Fuzzy semantic complex event FSCE****:*
A fuzzy semantic complex event is a semantic event (see Definition 10) $FSCE = \{t, G\}$ *where* t *is the timestamp, and* G *is a set of RDF-Triples forming a graph.* G *carries a fuzzy context data* fcd *and extra information.*

Such event is said complex as it results from a combination from multiple events. As mentioned, the fuzzy context data itself is a RDF sub-graph composed of triples sharing the same subject and predicate, and annotated with truth weight. Extra information carried by semantic events in SE_{cd} are also added in the graph G of the FSCE. However, now irrelevant information, such as confidence values and sources, are removed. An example of a FSCE resulting from the fuzzification of the batch in Figure 8.5 can be found in Figure 8.6.

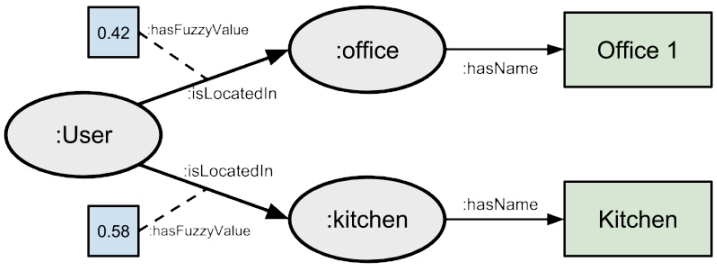


Figure 8.6: Example of generated FSCE

Example of a FSCE carrying a fuzzy context data with two values or objects, in grey. Trust are annotated to each possible value, in blue. They are associated with extra information, in green.

One FSCE is generated for each entering semantic batch. These FSCE are the output of FSCEP and can then be inserted in a context ontology, or directly used. Thanks to their semantic formatting, their integration is instantaneous. By using a fuzzified representation, **accuracy** and **contradiction** is modelled and can be used in cognition processes. On top of, **freshness** and **imprecision** were tackled in the previous phases of the generation.

8.3.4 Context Ontology

Once generated, the FSCE is sent to a context ontology that stores it. Note that it can be sent to another application, yet, in our work, we only considered a context knowledge. The provided FSCE is ensured to be fresh and precise, while contradiction and accuracy are modelled through truth weights. The fuzzyness of context data can then be used to perform adjustments of the context knowledge by reasoning over multiple context data. For example, if the fuzzy context data is provided `(user, isLocatedIn, {kitchen (0,58);bedroom (0,42)})` while another source provides `(user, isDoing, {cooking (1,0)})`, then the user is more likely to be in the kitchen, as cooking is often done there. Truth weights can also be used by other processes to take into consideration the uncertainty, for example, activity recognition tools, such as AGACY [161], can be more accurate by using such values. This is what we measured in our experiments to evaluate the relevance of FSCEP.

8.3.5 Experiments

In order to evaluate FSCEP, we set up a simple activity recognition scenario. In this scenario, the user, Arthur, is working in the office; we aim to identify this activity from context data provided by imperfect sensors. To do so, we need to acquire context data about his position and his current stance. The office is equipped with a beacon, a motion sensor and the chair has a pressure sensor, while the living room has a camera and a motion sensor. In this experiment, an activity is recognized by applying simple inference rules in a context ontology filled by FSCEP or a CEP without fuzzification. The objective of the experiment is to evaluate the gain of using a FSCEP over a solution that does not tackle uncertainty. In consequence, we measure the activity recognition rates with and without FSCEP.

Implementation

Figure 8.7: Screenshot of the Freedomotic interface with our custom devices

FSCEP was prototyped relying a well-known CEP, Esper [50], and ontologies [151] through the Jena framework. It is developed in Java and uses an agent based approach through Jade[4]. Experiments were conducted through simulation using Freedomotic, as discussed in Chapter 15. We have implemented our scenario and added multiple imperfect sensors, including motion sensors, beacons and cameras, as illustrated in Figure 8.7. These newly added devices generate events and are able to communicate with our FSCEP.

Protocol

500 runs were executed, with and without FSCEP, thus totalling 1000 runs. One run consists in a salve of events, the event processing through FSCEP or CEP, and the recognition of activity. We considered one activity per run. To model uncertainty, each sensor has a probability of providing an incorrect context data, ranging from 0% to 60%. In one run, for each sensors, context data are generated within Freedomotic, correctly or incorrectly. These virtual sensors communicate context data carried in events to FSCEP. These events are formatted to be a compatible with Esper. Formally, Esper is associated with a list of possible events, that is a list of Java classes, entering events are Java objects based on these classes[5]. We designed two inference rules to detect activities "working" or "resting", while Arthur is supposed to be only working in the scenario. We then measure the rate of successful activity recognition, meaning detecting that Arthur is working, with and without FSCEP.

Results

The result of correct activity recognition comparison is displayed in Figure 8.8. As we can see, by using FSCEP, the rule-based activity recognition process gains 10% of correctness. In fact, as expected, by having all possible values for a context data associated with truth weight, the cognition process can consider all options and associates each outcome with a confidence value. This proves the potential of our model for such applications. Note that, as home environments are each different, the gain varies accordingly. It is actually complicated to quantify an exact gain, as the uncertainty

[4] http://jade.tilab.com/

[5] http://esper.espertech.com/release-5.5.0/esper-reference/html_single/index.html#event_representation

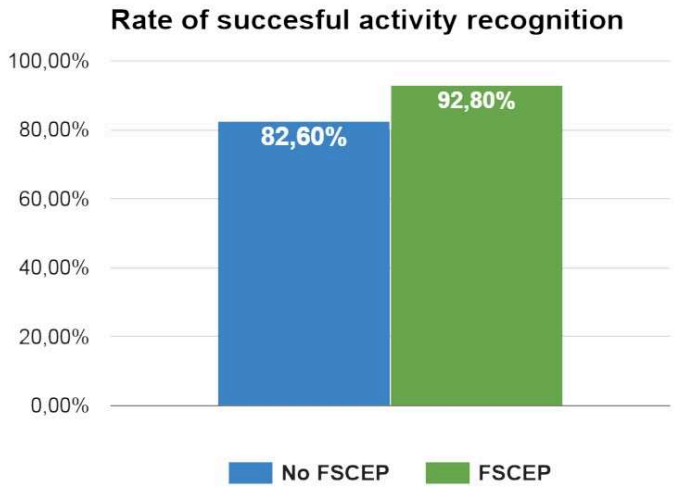


Figure 8.8: Result of FSCEP experiments

of each environment can hardly be quantified and varies according to the context and available devices. Hence, the strength of FSCEP is to enable other tools to perform their role more accurately, as we have shown with this simple rule-based activity recognition case. Yet, using events from sensors to acquire the context is not the only solution for the robot. In fact, it can use its cameras, which are however not perfectly accurate. This is the object of the next section.

8.4 Vision-Based Context Acquisition: VARSeR

Although the robot can rely on sensors, from itself or the smart environment, that provide events or event like data, not all devices are working this way. This is particularly the case of cameras, especially robot ones. Indeed, unlike sensors that measure a particular value, such as a temperature or the occurrence of the movement, cameras provide images, that, as is, do not provide information. In fact, images need to be analysed through vision algorithms. For a robot, vision is essential as the camera remains its main sensor. Consequently, vision algorithm, using both 2D and 3D images, have been extensively used and improved, in particular with machine learning techniques.

Even if user activities can be inferred from the analysis of the context knowledge, as performed with AGACY [161, 162] for example, the robot can also use its camera to do so. By analysing video streams, the robot can identify the gesture of the user and associate it to an activity. Combined with the mobility of the robot, such a tool enables the possibility to monitor the user anywhere the robot can go. Hence, the activity of the user can be known, even if it is located in a "blind spot" in the smart environment. However, such a technique is heavy, requires a strong learning phase and is subjected to **imprecision** and **inaccuracy**. Indeed, the output of the vision activity recognition can be erroneous, this can be due to a weakness in the learnt model or a confusion. Moreover, the gesture may not be enough to identity a precise activity, for example, vision may detect that the user is sitting, but not what for he is sitting, such as reading or watching the television. As the recognition is based on gesture, some of them are alike, making them hard to distinguish some of them, even for human. An illustrative example can be found on Figure 8.9. However, in a smart environment, the robot

is not alone and can count on other sensors to support it. The idea is that some data sources can provide determinant information. Furthermore, this offers the possibility to compute more precise activities. For that reason, we proposed a solution in that direction.

Visual Activity Recognition with SEmantic Reasoning (VARSeR) is an hybrid approach combining a learning vision-based activity recognition and ontologies. Our contributions improve a state-of-the-art vision-based activity recognition thanks to an ontological context knowledge. This leads to a solution that can be applied anywhere the robot can go, that requires few specifications as it uses learning, and that used a rich context knowledge to refine the recognition. It also stores all possible activities and rules, that are essential as we will see, in that sense, it can be seen as a background ontology. In VARSeR, the ontology is used, through rules, to correct and refine the output of a previously proposed vision-based process [46, 158] through rule based reasoning. Consequently, VARSeR is able to identify more specific and accurate activities.

Before presenting our hybrid approach, let us first briefly describe the used vision-based activity recognition process and illustrate its limit.

8.4.1 Vision-Based Activity Recognition

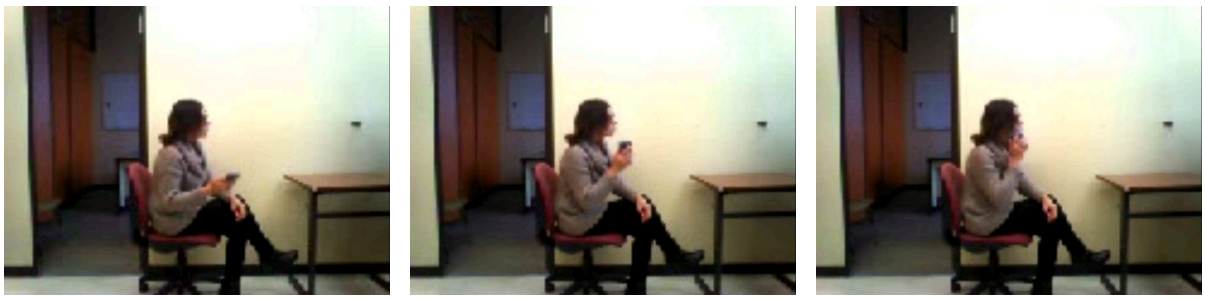


Figure 8.9: Example of a video stream from a Nao robot

Example of a video stream of a confusing gesture acquired by a Nao. Is the user phoning or drinking?

Our contribution is to improve a state-of-the-art vision-based activity recognition process by refining its result. Recognizing an activity aims to extract one or multiple activity defined as follows:

Definition 17. ***Activity***
An activity is a high-level context data (see Definition 10), thus a RDF-Triple (subject, predicate, object) where where $subject \in$ **Person**, $predicate = isDoing$ **and** $object \in$ **Activity.**

For example, `(:user, isDoing, :cooking)` is an example of context data representing Arthur, the user, being cooking. We suppose a user can only perform a single activity at a time. In this work, we used the proposition of El-Yacoubi et al. [46, 158, 120]. This approach consists of analysing a 2D video stream, as seen in Figure 8.9, acquired by the robot to classify the gesture of the user to possible activities. In brief, it uses as input a short **video stream** and outputs an ordered **list of possible activities associated with probabilities**. It is based on machine learning and requires a training dataset. This solution is a four-steps process:

1. **DenseTrack** [181] is an algorithm that analyses an image stream to provide pixel trajectories.

2. **K-means** [109] algorithm is used to group the trajectories into a fixed set of prototypes or clusters.

3. **Bag of words** algorithm is used to compute the frequency of 'visual word' thanks to the clusters generated by K-means.

4. **Support Vector Machine (SVM)** [72] The bags are classified using a trained SVM.

The algorithm can be trained by providing a data set consisting of annotated video example. Once trained, it can be applied on any acquired video stream. It generates an array of pair between activity and probability, for example: $\{(eating, 0.05), (phoning, 0.1), (walking, 0.01), ...\}$. We refer to this set as **activity distribution**. In our work, the algorithm was trained to recognize nine activities: *falling, phoning, applauding, drinking, remote control, sitting, walking, opening door, closing door*. Note that some activities are imprecise, for example *sitting*, as the user may be eating or watching the TV while sat. This process is heavy and time consuming, in fact, in our experiments, the robot needed 1 to 2 minutes to infer the activities for a 2 seconds gesture. It achieves decent result on dataset used in the literature, however, in more general cases, it encounters some difficulties and is sensitive to confusion between gesture, as illustrated in Figure 8.9. For an application for a personal robot at home, it needs to be enhanced, leading to our contribution. Please note that it would be possible to replace this activity recognition approach by another solution, as long as it provides a probability distribution of activities.

8.4.2 Refining Visual Activity Recognition

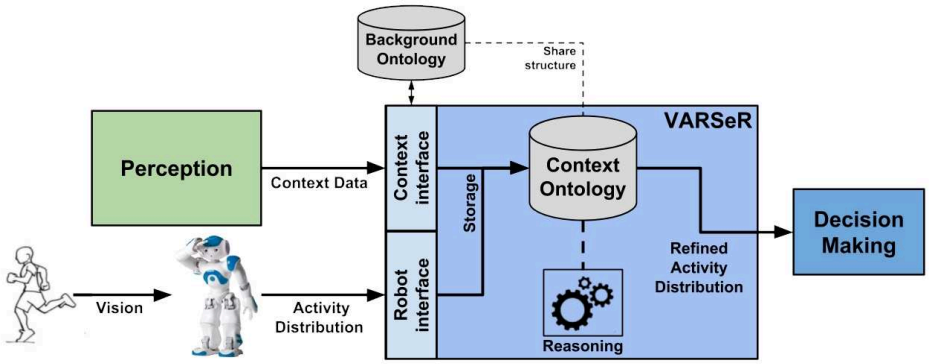


Figure 8.10: VARSeR process for activity recognition.

To overcome the limits of the vision-based activity recognition approach, VARSeR combines it with ontological context knowledge. VARSeR is composed of four main steps, illustrated in Figure 8.10:

1. **Vision-based activity recognition:** The robot looks at the user's gesture and captures a short video. It then analyses the video and classifies it into activities using the approach presented in the previous section. It sends the result, i.e. activities associated with probabilities, to the context ontology.

2. **Acquiring context data:** Once activities have been inserted in the context ontology, it starts the refinement process and gathers context data. These context data are formatted and stored in the ontology. Context data are pushed to VARSeR.

3. **Reasoning:** After acquiring context data, including activities, in the context knowledge, reasoning is applied. The reasoning aims to (1) infer new context data and (2) to identify more precise activities.

4. **Refining activity distribution:** By using the knowledge carried by the context ontology, probabilities are adjusted, thus correcting the inaccuracy of the initial vision-based recognition.

VARSeR aims to be fully deployed on a robot, however, it can also work with a robot and a server. In that case, the robot proceeds with the initial activity recognition while the server carries the context knowledge and performs the refinement. Let us now review each step:

Vision-based Activity Recognition

Our contribution comes on top of a vision-based activity recognition approach. Thus, the very first step is to apply this process. The robot observes the user to acquire a video of the gesture and use the algorithm presented in the previous section. From this, an array of activity associated with probabilities is provided and send to VARSeR. The array is formatted to ontological format by an interface and using the background ontology. The activities, associated to their probability, are then stored in the context ontology. As example, the following activity distribution may be provided and stored in the context ontology:

$$\{(sitting, 0.3), (phoning, 0.2), (walking, 0.05)\}$$

Acquiring Context Data

Once the vision process is over, VARSeR gathers the context data. To do so, it acquires and stores context data provided by sensors during a fixed duration. The data are formatted and pushed to VARSeR through specific interfaces. Note that we suppose their structure to be matching the ontology one, this is enabled by the background ontology that shares its structure with the context ontology. FSCEP can be used to provide these context data, but also other tools; VARSeR is actually independent to the context provider. All the received context data are stored in the context ontology that now carries the knowledge of the context.

Reasoning

Once filled, reasoning can be applied on the context ontology. Reasoning is performed by applying inferences rules. These enables to infer new context data, but also and mainly to refine activities into more precise one. In fact, detected activities can be imprecise, such a *sitting*. Yet, by using more data on top of the initial recognition of the robot, it is possible to transform activities toward more specific ones. For example, *sitting* can be refined into *watching TV* or *eating*. For each refined activities is associated to the same probability as the original activity. Inference rules provided by

the expert can update the activity distribution stored in the context ontology. Let us considered the following rules as example. They are written using Jena[6] formalism, but other inference language can be applied. Note that real rules are much more complex and discriminative.

- **If** the user is *sitting*, located in a room with a TV **then** the user is "watching TV":
  rule1: (?act is-a Activity) (?act a_label "sitting") (?act a_proba ?prob) (?usr isLocatedIn ?room) (?room is-a RoomTV) $\rightarrow$ *(?newAct is-a Activity) (?newAct a_proba ?prob) (?newAct a_label "watching TV")*
- **If** the user is *sitting*, and close to a table **then** the user is "eating":
  rule2: (?act is-a Activity) (?act a_label "sitting") (?act a_proba ?prob) (?usr isCloseTo ?tab) (?tab is-a Table) $\rightarrow$ *(?newAct is-a Activity) (?newAct a_proba ?prob) (?newAct a_label "eating")*

Hence, by supposing the user is sitting at the table in the living room, by applying these two rules, the activity distribution obtained by vision will be refined into:

$$\{(watchingTV, 0.3), (eating, 0.3), (phoning, 0.2), (walking, 0.05)\}$$

Note that these more precise activities are not necessarily dependent of the original activity, for example, the user may be eating while standing and can also be detected by other rules or sources. As we can see, the activity distribution can be ambiguous. Furthermore, as stated, the accuracy of the vision-based recognition is questionable. Thus, the activity distribution should be refined to lift the uncertainty.

Refining Activity Distribution

In order to correct the probabilities, the context is analysed and activities are adjusted accordingly. This is done by applying inference rules. For example, the following rules can be applied.

- If the phone's inertial unit exceeds a given threshold – increase "phoning" probability,
- If a movement was detected through a motion sensor located in a room with a TV – increase "remote controlling" probability,
- If the TV is switched on – increase "watching TV" probability.

These rules, as well as the probability adjustment, are defined by the designer. Obviously, further rules can be added, even when the system is deployed. By applying these rules, the firstly computed activity distribution is adjusted and corrected according to the observations of the environment. Typically, in case of a confusion by the vision process, a disambiguation is enabled by these rules. For example, let us considered that the phone moved during the gesture, after applying rules, the resulting activity distribution:

$$\{(watchingTV, 0.3), (eating, 0.3), (phoning, 0.5), (walking, 0.05)\}$$

Hence, the phoning is the now the most probable one. If not refined, the activities watching TV or eating would have been believed to be the one occurring. This refinement leads to a more **accurate** activity recognition.

[6] jena.apache.org

Outcome

In the end, VARSeR have collected activities from a vision-based approach and context data from various sensors, and have provided a new set of activities. These activities have had their probability corrected in accordance to the environment, and, on top of that more precise activity were recognized. This enables a more precise and accurate understanding of the activity of the user, as we observed through our experiments that we discuss in the next section.

It is worth noticing that the vision-based process, as well as VARSeR, only consider one activity at a time. In fact, the activity distribution is built assuming only one activity to be true. However, in reality, multiple activities can be performed at the same time. Typically, the user may be watching TV while eating. An interesting perspective would be the ability to recognize multiple activity at once. A way to approach it is to rely on fuzzy logic and truth weights rather than probabilities. Note that, we considered only one activity at a time in our experiments.

8.4.3 Experiments

In order to evaluate VARSeR, we conducted extensive experiments within the HadapTIC platform and with a Nao robot. In those tests, we aimed to evaluate the performance of the vision-based solution by itself, and the performance of VARSeR, that combines vision and ontological approach. These tests consisted of volunteers doing gestures in a smart room and in front of a robot that tries to identify his/her activity. A detailed discussion of these experiments can be found in Appendix A, in this section, we only focus on the important results.

Implementation

In our tests, we used the vision-based activity recognition proposed by El-Yacoubi et al. [46] that was already implemented. It was embedded in a Nao robot. The HadapTIC platform provided multiple sensors, including motion sensors and smart phones.

Protocol

Figure 8.11: Photo of VARSeR experiments for a "remote control" scenario

The experiments consist of volunteers performing gestures in front of the robot, as shown in Figure 8.11, the vision process and VARSeR are then applied and evaluated. We considered three scenarios that were repeated multiple times for each volunteer: (1) Make a phone call, (2) Activate the remote control and (3) Open the door. From this, we measured the correctness of the recognition with and without VARSeR.

Results

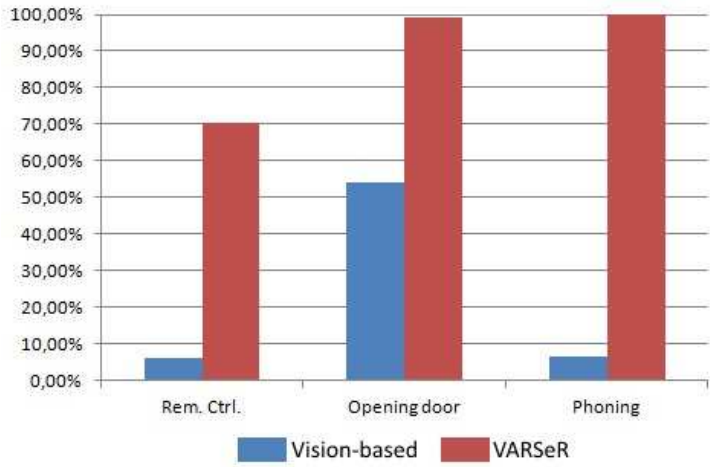


Figure 8.12: Successful activity recognition rate per scenario

Figure 8.12 shows the activity recognition success rate with and without refinement. Firstly, as we can see, raw vision-based activity recognition had mediocre results. It achieves 6.40% for the "remote controlling" scenario, 53.97% for the "opening door" and 6.35% for the "phoning" one. This result points out the difficulty for the vision-based algorithm to have a clear and strong classification by itself. Indeed, the vision process is highly subjected to confusion, as gesture were often mistaken. Typically, the remote-control gesture was often confused with an opening door and applauding gesture, leading to a poor recognition rate. However, as for the phone scenario, the activity recognition often detected the phone gesture as the lowest probable activity. We assert the cause of this poor result to be the lack of training, even if previously trained using datasets, which points out the need for an extremely strong learning phase. These results underline the need for an improvement.

On the other hand, results provided by VARSeR, that uses the outcome of the vision process and context data from smart devices, are much better. The refined "remote controlling" scenario has 71% successful recognition rate while the two others were almost reaching 100% of recognition with VARSeR. This is again explained by the impact of the sensors. These results have also to be put into perspective: in a more complex scenario with more activities and data sources, these rates would not be equally good. Such high results are also and mainly explained by the semantic carried by those context data: for example, a phone being lifted, detected through its inertial unit, provides a strong hint toward the phoning activity. On top of that, as the activity distribution of the vision process (see Appendix A) is tight, even a minor correction of the probability distribution allows to provide a correct result. Hence, the information provided by external sensors have more weight in

the recognition. This questions the relevance of using such a combination. Nevertheless, results also show the gain we can have by combining and/or relying on smart sensors.

To summarize these experiments, we can assert that combining vision and sensors from the environment allows to provide more reliable results. But, our main conclusion is that using both approaches at the same time is not always the best solution. In some cases, the vision process could be avoided, preventing huge costs in time and resources. In fact, we believe that **selecting a method to recognize activities** instead of combining them in any case would be a better solution. Typically, in this case, the robot could select either the vision, the smart environment, or both of them, depending on the current case. This is obviously a challenging, yet promising, issue to tackle.

8.5 Conclusion

In this chapter, we proposed new techniques for context acquisition for a robot in a smart environment. We considered two types of sensors: (1) event-based or event like sensors, that provide specific measurement of the environment, such as the temperature or the occurrence of a movement, and (2) cameras, in particular robot ones, that provide a video stream and requires a vision algorithm to extract information. Hence, we proposed two contributions that support the constraints of robots within smart environments, in particular uncertainty.

Our first approach, FSCEP, combines CEP, ontologies and fuzzy logic to acquire, filter, semantize and fuzzify events. FSCEP uses CEP rules that allow filter and create batches of event, hence coping with *freshness* and *accuracy*. Then, by relying on a background ontologies it enriches events with further data, solving the *imprecision* of events and associating them with a confidence value. Finally, FSCEP analyses these enriched events to generate a single Fuzzy Semantic Complex Event that carries fuzzy context data whose truth degrees are computing thanks the added confidence value. By doing so, the resulting event models both both **inaccuracy** and **contradiction**. Consequently, FSCEP provides context data that are freed from uncertainty constraints and that carry multiple possible values associated with a degree. Thanks to this handling and computing of truth degree, FSCEP enables more accurate cognition processes that can use the provided data. This is what we observed in our experiments, were a simple activity recognition solution was improved by taking into consideration various possible values and uncertain degree.

Our second contribution, VARSeR, is a hybrid approach that combines a vision-based process and ontologies for recognizing user activities, that are context data. Indeed, by itself, and as underlined by our experiments, the vision-based approach is inaccurate and imprecise. By combining these results with context data provided by other sensors and using ontological reasoning, VARSeR refines the detected activities through rules. Based on the context knowledge, the refinement rules actually adjust the probabilities of activities resulting from the vision process, hence correcting its inaccuracy. Furthermore, inference rules are used to infer more precise activities, based on observed gestures and observations from external sensors. Our results show VARSeR to achieve more accurate and more precise recognitions. However, our experiments also question the relevance of using both approach at the same time rather using one or the other according to the availability and reliability of sensors. In fact, context data provided by sensors appeared to be discriminative enough,

while the vision process could not provide strong distinction between activities. On top of that, in this contribution, only one activity is considered at once.

An interesting perspective would be to adopt a fuzzy rather approach than a probabilistic one in VARSeR. By doing so, it would be able to identify multiple activity at the same time. Exploring the balance between learning based and specification based for activity recognition is also an open door for future work. Nevertheless, our contributions enable a context acquisition under uncertain sources. Our approaches can also be easily extended with new rules, and support video streams and various types of events. The resulting context knowledge can then be used to infer new data and take decision while supporting uncertainty. In the next chapter, we discuss how the robot can detect anomalous situations and intervene consequently.

Chapter 9

From Knowledge to Decision

What makes a personal robot truly intelligent is its ability to think and take decisions on its own. After perceiving the context through its sensors or smart devices in the environment, the robot applies cognition techniques to understand the acquired context knowledge and, consequently, to decide what to do accordingly. This includes the understanding of situations, in particular anomalous situations. Such situations are non-ordinary, thus anomalous, and non-acceptable state of the environment where the robot should intervene. These are not necessarily hazardous situations, that can be detected through specific sensors, but rather situations where a problem could occur. In other words, detecting anomalous situations allows the robot to act before the problem occurs. For example, in our scenario, Arthur may forget he is cooking and go to sleep. In such situation, as the oven is unattended, there is a risk of smoking and/or fire. By understanding the situation, Nono can act to switch off the oven or wake up Arthur before the actual problem could happen. Doing so is naturally more complex than reacting to the problem itself, that could have been simply detected through a smoke detector in this example. In fact, in order to understand the situation, the robot needs to continuously analyse the whole knowledge of the context, including the activity of the user.

However, in the literature, most works only consider anomaly over activities or specific data. In fact, an anomaly is often consider to be a derivation of a usual routine of activities [75, 147, 79] or a derivation on designated data [106, 125, 126]. Even if multiple axis were explored, with both learning based [70, 78] and specification based [146, 147, 5] techniques, none of them is able to identify anomalous situations over the global context knowledge. Hence, no technique is for example able to detect the anomalous situation of Arthur sleeping while he started cooking. That is why we explore the recognition of such situation, that is essential for a personal robot at home.

A simple way to identify anomalous situations over various context data is through the specification of rules. For example, a rule can state that is if the oven if on, the temperature in the kitchen is high, and that Arthur is sleeping, then the current situation is anomalous. However, the world is far from being perfect. In fact, sensors are imperfect, leading to uncertain context data that makes rules inefficient. The context knowledge can be outdated, inaccurate, imprecise, in contraction or incomplete. As we discussed in the previous section, to tackle uncertainty when acquiring the context, we proposed a tool, FSCEP, that tackles and models uncertainty through fuzzy context data. We also proposed VARSeR, that provides a visual activity recognition solution with improved accuracy and precision. Furthermore, the context knowledge maybe incomplete, meaning some context data

maybe missing. For example, in our example, the temperature in the kitchen maybe inaccurate while the activity of Arthur unknown, as Nono could not see him; nevertheless, Nono should understand that the current situation is anomalous and intervene. In fact, even if the knowledge is uncertain, it is important for the robot not to misunderstand those situations, as they could lead to problems or hazards if no interventions are performed. That is why we explore the usage of a hybrid approach: Markov Logic Network (MLN), a combination between Markov network and first order logic. Through this technique, we aim to enable rules evaluation under uncertainty.

In this chapter, we present Context and Activity Recognition Enabling Detection of Anomalous Situation (CAREDAS). CAREDAS is cognition approach that uses ontological knowledge and MLN to identify anomalous situations. Rules are modelled through MLN that enables to use modelled uncertainty and to cope with completeness: an anomalous situation can be recognize over uncertain context data and even if not all elements of a rule are observed. By using a context ontology filled using perception techniques previously described, CAREDAS is able to use the whole knowledge of the context. Moreover, it performs a continuous analysis of the context by using time windows. In the end, thanks to a mapping, the robot decides of action to perform to cope with the current situation. Hence, CAREDAS use an uncertain knowledge to achieve a correct decision.

Before describing CAREDAS itself, we first have a preliminary discussion around MLN and used notions. We then describe how CAREDAS uses ontology and MLN to identify anomalous situations and take a decision. Finally, we present our results based on a dataset we acquired in a smart environment.

9.1 Preliminaries

As CAREDAS relies on MLN, let us first discuss this technique. We then set up definitions of notions in this work.

9.1.1 Markov Logic Network

Markov Logic Networks [149] are a combination between Markov network and first order logic that enables uncertain inference.

Before defining MLN, it is important to know what is a Markov Network. A Markov network is an undirected graph model for the joint distribution of a set of variable $X = (X_1, X_2, .., X_n)$. It is composed of a set (F_i;w_i); $1 \leq i \leq n$, where each F_i is any real-valued function of the state and $w_i \in \mathbb{R}$ is its weight. Z is the partition function. The joint distribution is given by:

$$P(x) = \frac{1}{Z} exp\left(\sum_i w_i F_i(x)\right) \tag{9.1}$$

MLN are formally defined as follows by Richardson & Domingos [149] :

Definition 18. ***Markov Logic Network:***
A Markov logic network L is a set of pairs (F_i;w_i), where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants C = c_1...c_n, it defines a Markov network $M_{L,C}$ (Equation 9.1) as follows:

1. *$M_{L,C}$ contains one node for each possible grounding of each predicate appearing in L.*

2. *$M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L.The weight of the feature is the w_i associated with F_i in L.*

A MLN can actually be seen as a template for building a Markov network: these networks are called *ground Markov networks* (GMN). The resulting ground Markov network is a graph where nodes are predicates. A vertex associates two nodes if the corresponding predicates appear together in at least one grounding of one formula in L. From Definition 18 and Equation 9.1, the probability distribution in the ground Markov network $M_{L,C}$ is given by:

$$P(x) = \frac{1}{Z} exp\left(\sum_{i}^{F} w_i n_i(x)\right) \tag{9.2}$$

Where F is the set of rules in the MLN, $n_i(x)$ is the number of true groundings of F_i with constant $x \in C$, w_i is the weight of F_i, and Z is the partition function.

The generated GMN, thus graph, can then be analysed to evaluate formulas or rules. Through the weight and resulting probability distribution, MLN allows to evaluate those rules under uncertainty Note that MLN actually uses specification as for rules and structure, and learning as for weights, consequently, it can be considered as a hybrid approach. Please refer to Richardson & Domingos work [149] for a more detailed definition.

9.1.2 Definitions

CAREDAS uses multiple notions. Firstly, it uses fuzzy context data that can be provided by FSCEP. For example: `(user, isLocatedIn, {kitchen (0,6);bedroom (0,4)})`. The notion of fuzzy context data is defined in Definition 13. We suppose each FCD to be associated with a timestamp t, that correspond to the time carried by a Fuzzy Event that carried this context data. This timestamp will be used for time windowing. FCD can represent any context information, including the user activity, provided by VARSeR or AGACY.

CAREDAS also considers the particular case of static context data.

Definition 19. ***Static Context Data (SCD)**:*
A Static Context Data is a context data where $subj$, $pred$ and val are not varying over time.

A SCD actually represent background information provided by an expert about the environment. For example, a SCD $(motion1, islocated, bedroom)$ that states that a given sensor is in a room is provided and does not vary.

Context data are stored and accessed through a context ontology. Note that context data can be represented in different ways. While represented as RDF triples by FSCEP and in the context ontology, MLN relies on predicate. Yet, the transformation from one format to another is trivial and is supposed transparent. For example, the location can be modelled as: `(user, isLocatedIn, {kitchen (0,6);bedroom (0,4)})` or `isLocatedIn(user, kitchen, 0.6), isLocatedIn(user, bedroom, 0.4)`.

The key notion of CAREDAS is the anomalous situation, that is, first of all a situation.

Definition 20. ***Situation:***
A situation is a triple: $(\{CD_i\}, t_s, t_e)$. $\{CD_i\}$ *is a set of context data captured during a time interval* $[t_s, t_e]$; t_s *is the start time of the execution of the* $task$ *and* t_e *is the end time of its execution.*

Hence, a situation can be seen as a 'snapshot' of the state of the environment. An anomalous situation is a particular situation were the robot should intervene. It is defined as follows:

Definition 21. ***Anomalous Situation:***
An anomalous is a particular situation associated with a class cl *and a confidence weight* w. *Hence, an anomalous situation can be seen as a triple:* (s, cl, w) *were* s *is a situation.*

Note that the class corresponds to a type of of anomalous situations, for instance, a risk of fire or an unreachable recharge station for the robot. Anomalous situations can be defined though rules. The principle is that, if all predicates, thus context data, of the rule are observed, thus the current situation is anomalous. For example, the rule: `isDoing(user, "sleeping",` w_a`)` $\wedge$ `status(stove, "on",` w_b`)` $\rightarrow$ `Anomaly(riskFire,` p`)` states that if the user is sleeping while the stove is on, therefore the situation is anomalous for a risk of fire.

9.2 MLN for Anomalous Situation Detection

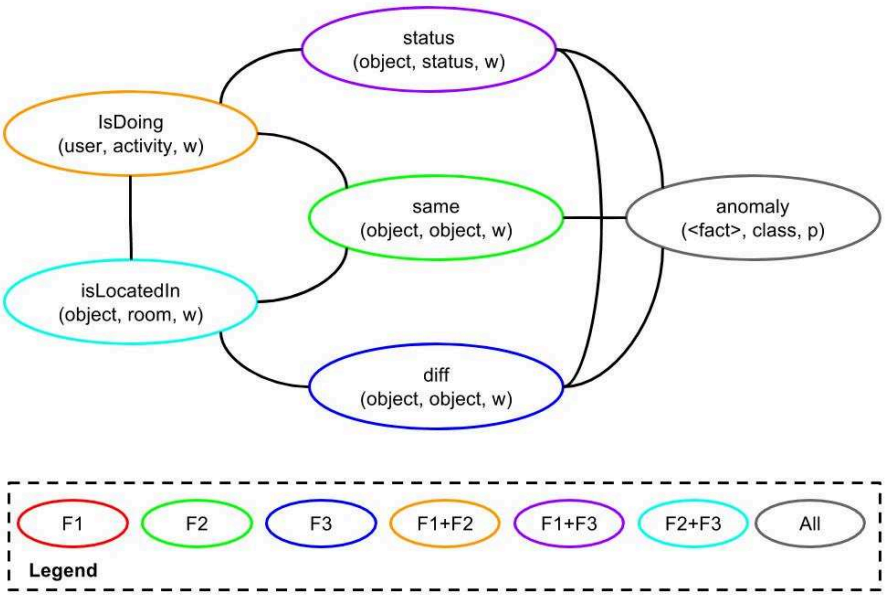


Figure 9.1: Example of a MLN with three rules

The colours indicate in which formula a predicate is used.

In order to recognize anomalous situations under uncertainty, these situations are defined by rules modelled in a MLN. A MLN is a non-oriented graph representing the dependencies between formulas and predicates, in other words, between rules and type of context data. Let us have an example of definition of anomalous situations, let us consider the following rules:

- F_1: isDoing(user, "sleeping", w_a) $\wedge$ status(stove, "on", w_b) $\rightarrow$ Anomaly(riskFire, p)
- F_2: isDoing(user, "sleeping", w_a) $\wedge$ isLocated(user, roomA, w_b) $\wedge$ isLocated(rechargeStation, roomB, 1) $\wedge$ same(roomA, roomB, 1) $\rightarrow$ Anomaly(rechargeUnreachable, p)
- F_3: isLocated(robot, roomA, w_b) $\wedge$ isLocated(rechargeStation, roomB, 1) $\wedge$ diff(roomA, roomB, 1) $\wedge$ isLocated(door, roomA, 1) $\wedge$ isLocated(door, roomB, 1) $\wedge$ status(door, "close", w_f) $\rightarrow$ Anomaly(rechargeUnreachable, p)

The MLN based on these rules is represented in Figure 9.1. As we can see, the rules rely on particular predicates that are derived from both FCD and SCD We define three types of predicates:

Definition 22. ***Probabilistic Evidence Predicates (PEP)**:*
Probabilistic Evidence Predicates (PEP) in our MLN is a predicate corresponding to FCD. It is carries an evidence value that is the weight w.

Note that, when initiated from a FCD, $|mVal|$ PEP can be generated, in other word a predicate is generated for each possible value. The weight w is set to be the truth value w_v corresponding to the value v.

Definition 23. ***Deterministic Evidence Predicates (DEP)**:*
Deterministic Evidence Predicates (DEP) in our MLN is a predicate corresponding to SCD. A DEP is always true.

Hence, $isLocated(rechargeStation, roomB, 1)$ is an example of a DEP representing the FCD of the location of the recharge station.

Definition 24. ***Probabilistic Hidden Predicate (PHP)**:*
Probabilistic Hidden Predicate (PHP) in our MLN can be referred as the query node: the right side of the rule. Its distribution truth p is unknown.

Anomaly(AnomalyClass, p) is one of the PHP considered in CAREDAS. The objective of our MLN, thus CAREDAS, is to infer p for each PHP in the graph. This value represents the confidence of the rule being true, thus it represents the confidence the situation to be anomalous with the associated anomaly class. In consequence, computing and evaluating p for all rules allows to establish the abnormality of the situation. Let us now see how p can be computed.

9.3 Anomaly Detection

The main role of this step is to find the most probable anomaly situation class, if any, of the current situation. To do so, three steps are required:

- Situation construction: the situation is constructed from the context data provided in time window. The process of anomalous situations is performed according to these time window

- Rules weights calculus: based on the instantiated formulas, weights of formulas are computed and the MLN is used to compute probabilities of constant.
- Hidden Predicates (PHP) weights calculus: computation of PHP based on probabilities of constant and detection of anomaly.

9.3.1 Situation Construction

CAREDAS relies on a context ontologies that is continuously fed through tools such as FSCEP, VARSeR or AGACY. Entering context data are fuzzy, hence associated with truth values. Activities are also associated with a weight corresponding to the confidence in the recognition. technically, any context ontology can be used, for example the previously presented context ontology for human activity representation [151] can be used. Inference rules can be applied to infer new context data. The resulting ontologies carries a rich and various knowledge of the context, for example, user location, activities, topology of the environment or robot battery level are data stored in the ontology. Having a large and rich context knowledge is important to have a precise understanding of the situations based on the whole context. From this knowledge, instance of FCD and SCD are extracted and then used to generate a GMN from MLN, as we will see in the next sections.

Situations are considered within fixed time window. A time window is a period of time in which context data are gathered. Meaning all SCD and all FCD whose associated time is inside the time window are considered. These context data are then used to instantiate the rules carried in the MLN. At the end of the time window, these rules are used in the recognition process. However, the instantiated formulas are not removed at the end of the time window. In fact, each time window enriches the current knowledge. The idea is to keep enriching the set of instantiated rules to refine the recognition process: the more time windows pass, the more instance can be used to identify anomalous situations. If an anomaly is detected, the activity of the user changes or an arbitrary duration passed, the inference is reset and the instantiated formulas are purged.

Let us now see how the extracted context data are used to instantiate rules and identify the anomalous situations.

9.3.2 Formulas Weights Calculus

Once the context data included in a time window are gathered, they can be used to instantiate the MLN into a *ground Markov network*. Extracted context data provide a set of **IPEP**$_w$ (weighted **I**nstanciated **P**robabilistic **E**vidence **P**redicate) where :
$\forall$ instP$_w \in$ IPEP$_w \leftrightarrow$ instP$_w \in$ InstP & the truth value of the predicate is known. For example, can have to instantiated predicates: status(oven, "on", 0.95), isLocated(user, "livingroom", 0.25) isLocated(user, "bedroom", 0.75) are IPEP$_w$ obtained from FCD. Then, to compute the weight (w_i) of logic rule $F_i \in$ F of the MLN model we apply this proposed formula :

$$w_i = ln \frac{(\sum_j w_{satP_{ij}})/n_i}{1 - \sum_j w_{satP_{ij}}/n_i} \tag{9.3}$$

Where satP$_{ij}$ $\in$ IPEP$_w$ & satisfies F$_i$. w$_{satP_{ij}}$ is the weight of satP$_{ij}$. n$_i$ is the number of predicates $\in$ PEP $\bigcap$ F$_i$. This weight determine the confidence of a formula being valid, even if not all predicate of this formula are observed. This modelling enables coping with incompleteness. For example, the F_1 will have the weight: $w_1 = ln\frac{0.95/2}{1-0.95/2} = -0.1$.

These weights can then be used in MLN Equation 9.2 for each constant. By applying this equation and instantiating nodes, the GMN can be generated from the MLN. This graph is then used to infer the confidence of PHP, leading to the recognition of the anomalous situations.

9.3.3 Computation of the weight of Probabilistic Hidden Predicates

Once the GMN is generated, CAREDAS can now compute the weights of the PHP. By doing so, each possible anomalous situation, modelled in the rules, will be associated with a confidence value. Let k be a PHP in the GMN, P$_{pk}$ is its confidence weight p . Let C_k be the of all constants that are in predicates which have an edge toward k. P is the formula in Equation 9.2.

$$P_{p_k} = \frac{\sum_{c \in C_k} P(c)}{|C_k|} \tag{9.4}$$

In the end, all PHP have their distribution truth p computed. This value is used to establish if an anomalous situation class is likely to be occurring. If a PHP has a p beyond a given threshold, at default 0.5, the current situation is considered anomalous with the class of anomaly associated to the PHP. If multiple anomaly classes are detected, the one with the highest p is considered to be the most relevant. Lastly, if the situation is anomalous, the decision making process is started.

9.4 Decision Making from Anomalous Situations

Once an anomaly has been detected, it shall now be taken into consideration. In our context, when facing an anomaly, the robot should act to inform the user or solve it. Thus, whenever an anomalous situation is detected, the robot decides a new goal, it then will use its planner to generate a sequence of task to risk this goal. These goals aim to provide a solution to the situations according to the anomaly class detected. It can be simple, for example, if Nono can't access its recharge station, it decides to sleep to save battery. On the other hand, Nono may need to perform more complex task, like opening a door then moving to Arthur in order to warn him. This decision process is determined by a reaction knowledge RK defined as follows:

Definition 25. ***Reaction Knowledge**:*
The reaction knowledge RK is a set of couples (stc, g) where stc is an anomalous situation class and g is a goal to be executed by the robot.

As the anomalous situations are provided by an expert through rules, the reaction knowledge is provided alongside when designed. Thus, whenever an anomalous situation is detected, the robot access its reaction knowledge and finds the matching response. How the robot achieves its goal is the matter of the task planner, addressed in the next chapter.

9.5 Experiments

In order to validate CAREDAS, we conducted experiments using a dataset acquired in a smart environment. We aimed to measure the correctness of CAREDAS, as well as its precision and recall, when facing various situations. As CAREDAS relies on time window, we compared its behaviour with various size of time windows. In fact, a smaller time window size allows a quicker response, but also implies less context data to use, thus making the recognition harder. Let us review these experiments.

9.5.1 Implementation

CAREDAS was prototyped in Java and used Jena for ontology management. For this experiments, it was linked with a dataset we acquired. In a more general case, the prototype was integrated with a FSCEP implementation, for data acquisition, and AGACY activity recognition system.

9.5.2 Protocol

In order to evaluate CAREDAS, a dataset acquired out of more of 2 hours of daily life routine in the HadapTIC platform. The dataset carries a list of fuzzy context data, including user activities, associated with a timestamp. Before being provided to CAREDAS, each data of the dataset is transformed as a predicate, this step simply consist of creating the predicates in CAREDAS formalism, that is to say as proper Java objects. The 2 hours of acquisition can be divided in 15 instance of scenarios. For each instance, the tester performed multiple activities, triggering sensors in the platform. Activities were manually annotated. These scenarios aimed to recreate anomalous situations, but some of them didn't carry any anomaly. Each scenario was manually associated to an anomaly situation, that is the expected result of the identification process. By doing so, we aim to determine the precision and recall of CAREDAS. The dataset can be found online[1].

For this evaluation, a set of logic rules has been defined for multiple anomalies class. We have tested CAREDAS over multiple values of $win \in [60s, 300s]$. CAREDAS was evaluated by comparing its output against expected results. For each time window, the precision and recall of the system was computed. Furthermore, we computed the correctness, which is simply the rate of correct (expected) answer of the system. Figure 9.2 shows the obtained results and are discussed in the next subsection.

9.5.3 Results

Figure 9.2 shows the precision, recall and correctness of CAREDAS. As previously seen, these term are defined as follows:

- **Recall:** Number of anomalous situations detected, divided by the actual number of anomalous situations.
- **Precision:** Number of correct anomalous situations detected, divided by the total number of anomalous situations detected, correct or incorrect.

[1] `https://drive.google.com/file/d/1IqvO3dtpMR4gQkS4HjywSDoOTm1cC9dG/view?usp=sharing`

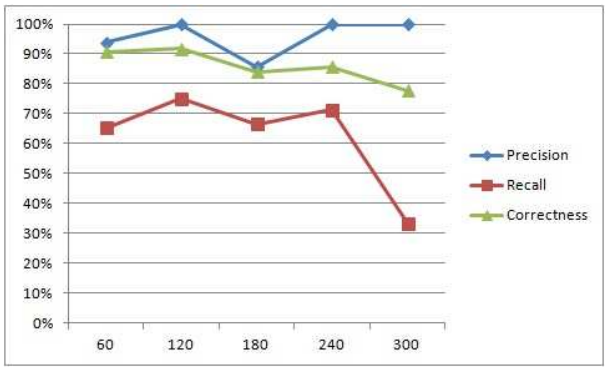


Figure 9.2: Precision, Recall and Correctness achieved by CAREDAS

- **Correctness:** Number of correct situations assumptions, divided by the total number of considered situations.

As we can see in Figure 9.2, CAREDAS has a high precision for all time windows. This means our solution is rarely mistaken when detecting an anomalous situations, which is important for user and robot safety. For shorter time windows, the recall is close to 70%. Even if it is a relatively decent result, it is not enough, as possibly risky situations can be missed by our approach. We can notice that after 4 minutes, the recall drops. This drop can be explained by the increase of data volume and the time window overlapping multiple activities, these two phenomena lead to confusions in CAREDAS. As the correctness, it is overall high, but decreases with the size of the time window, going from 90% of correct results with 1 minute time window, to 78% with 4 minutes time window. Yet, this shows CAREDAS to be globally matching the expected results of anomaly detection. In the end, we noticed CAREDAS achieve its best result with a 2 minutes time window.

All in all, our experiments, based a dataset obtained in the HadapTIC platform, shows CAREDAS to provide correct and precise anomalous situation identification. Thanks to its uncertainty support, in particular completeness through MLN, it is able to avoid mistakes in detection. However, it needs improvement, at it only enables a 70% recall at best, which is too low as situations can be hazardous. Finally, we noticed that 2 minutes times windows seems to be the best time windows size. However this value can be variable, as in other context with different sensors, the volume of context data may be different. Yet, as our tests were conducted in home-like environment relatively close to reality, we consider a 2 minute time window to be a decent default value.

9.6 Conclusion

In this chapter, we presented CAREDAS, a solution for anomalous situations detection using ontological knowledge and Markov Logic Network, a hybrid approach between first order logic and Markov models. CAREDAS models situations as formulas carried in a MLN that is instantiated into a Markov Network thanks to an ontological context knowledge. The ontology carries a rich context knowledge, with uncertain degrees, provided by previous contributions, that are VARSeR and FSCEP. In fact, CAREDAS takes advantage of the fuzzy context data provided by FSCEP and weighted activities given by VARSeR when creating the Markov Network. The resulting Markov Network can then

be used to evaluate situations. Thanks to the MLN approaches and the ontological knowledge, situations rules can be evaluated from an incomplete knowledge while using the previously uncertain degree. Consequently, the decision is made by supporting five dimensions of uncertainty: namely freshness, inaccuracy, imprecision, contradiction, that are tackled and/or modelled by FSCEP, and completeness, handled thanks to MLN. This enables an accurate recognition of anomalous situations[2] in uncertain environment.

To evaluate this assessment, CAREDAS was confronted to a dataset acquired in a smart environment were multiple situations were performed. Our results show our contribution to identify anomalous situations with a high correctness and precision, but also with an improvable recall. In fact, due to the amount of data, CAREDAS could sometimes be confused, this is underlined by the fact that the recall drops for larger time windows. Nevertheless, the decision is accurate, even when relying on uncertain data sources.

[2] Note that CAREDAS can actually be used to recognize normal situations as well.

Chapter 10

Conclusion

In this part, we presented three contributions that enable context acquisition and decision making. They are able to cope with the constraints encountered by robots in smart environments, namely heterogeneity, evolution, modelling, and, particularly, uncertainty.

Firstly, we proposed context acquisition tools for both event-based sources and video-based ones, particularly from robot cameras. The first is a specification-based combining CEP, semantic knowledge and fuzzy logic. It allows to filter, batch, semantize and fuzzify events into one fuzzy semantic complex event. By doing so, our approach is able to model and/or tackle freshness, inaccuracy, imprecision and contradiction. Furthermore, it supports various types of events and can be extended by adding new rules and knowledge. These features enable our technique to provide fuzzy context data that can be used to improve the accuracy of a cognition process. Indeed, our experiments show that our proposition could improve the results of a rule-based activity recognition solution thanks to its handling of uncertainty. Our second proposal is an hybrid approach for visual activity recognition. A learning-based vision algorithm, that recognizes user's gestures, was improved using a context ontology filled thanks to devices in the smart environment. Our proposal enables a more accurate and precise recognition, however, our experiments also question the relevance of such a combination. In fact, we observed that the vision algorithm enables a limited discrimination between activities, consequently, the sole usage of external sensors appeared to be enough to infer the activity. A convincing alternative would be to select which sources, video streams or events, to use for activity recognition.

Secondly, based of the context knowledge provided by the two first contributions, we proposed a technique to identify anomalous situations and make a decision for the robot. We designed a contribution that uses MLN, a hybrid technique based on first order logic and Markov models, and ontologies to accurately recognize situations where the robot should act based on a rich context knowledge. MLN is instantiated into a Markov network thanks to the context ontology, that carries the context knowledge associated with uncertain degrees. The model is able to take into consideration the uncertain degree provided by our previous contributions and to identify situations even under incomplete knowledge. This enables an accurate situation identification, even when relying on uncertain sources. This was validated by a confrontation to a dataset representing various anomalous and normal situations.

In the end, when assembled, these contributions allow to perform a complete abstraction of

data, from simple events to situations and decision making, while supporting the constraints of robots in smart environments, in particular uncertainty. Indeed, when associated, our techniques are able to handle the five dimensions of uncertainty, namely imprecision, inaccuracy, freshness, contradiction and incompleteness. On top of that, by relying on ontologies, they support a large variety of data and complex information, such as activities or situations. Furthermore, they can be easily adapted by adding new rules or new knowledge. Actually, these contributions form FAIRIE perception and cognition. With these tools, the robot can act knowing its decision is reliable and important to achieve. But how can it achieves its goal without failures?

Part III

Acting Without Failures

Chapter 11

Introduction

For a robot, it is not only essential to sense, but also to act on the environment. In fact, acting is what truly enables personal robots to achieve their roles, by helping users for instance. In order to act, robots use the context knowledge they acquired. This knowledge is already used in cognition for them to make a decision, but is also useful to establish how to achieve it. Task planners are a perfect tool for that matter. However, in domestic environments, robots can encounter task failures. This causes the plan to be altered and the reach of the goal to be delayed. Tasks failures can have various causes: the robot may have a break down, it may be limited in functionalities, a change may have occur in the environment, or the robot may be lacking proper planning knowledge. When failing, most techniques rely on replanning, that is to say recreating the plan. Yet, this is not a satisfactory solution as the robot does have to endure the failure and the reach of the goal is delayed. Even if uncertainty of context data can be an issue, handling failures due to the dynamism of the environment or lack of planning knowledge is a primordial issue. In fact, supporting uncertainty is irrelevant if the robot keeps failing for other reasons. That is why we first focus on failure issues and leave uncertainty management in planning for future works.

In this part, we present our contributions of the action phase, as well as cognition phase, of FAIRIE. Firstly, we proposed a novel planner based on HTN. Instead of planning and then execute the plan, our planning approach incrementally generates the plan and executes it in a same process. Furthermore, as for context acquisition, it relies on observation goals, that limits the usage of context data to the strict requirements of the planning. By dynamically generating the plan while unrolling it, our novel planner is able to take into account the last changes of the environment. This allows to prevent failures rather than reacting to it. However, as our planner relies on a rich planning knowledge, not all constraints can be foreseen, leading to failures. Secondly, we proposed a reinforcement learning technique to cope with that issue. This cognition contribution aims to enhance the planning knowledge from the experience the robot has acquired from previous tasks executions. From this experience, our approach identifies failure causes that can then be used by the planner to proactively prevent task failures. These two contributions allow to handle the issue of failures, as we will see in the two upcoming chapters.

Chapter 12

From Decision to Actions

12.1 Introduction

What makes service robots truly helpful is their ability to act. By being able to interact and act on the environment, these robots are able to fulfil their role of majordomo, guide or companion. To do so, robots can rely on popular tools that are task planners. Unlike static techniques such as FSM or BT, task planners offer a flexible way to achieve a goal, provided by decision making for instance. In fact, such a tool aims to generate a sequence of tasks for the robot, helped by the smart environment, to perform based on the knowledge on the context and an objective. For example, if Nono decides to warn Arthur about the forgotten active oven, it will analyse the context and create a sequence of actions that allows it to reach Arthur and to alert him in the most relevant way. Typically, if Arthur is sleeping the plan may include tasks to wake him up before delivering the message. However, in a smart and domestic environment, constraints are particular and task planners encounter limits. As we pointed in Chapter 5, the state-of-the-art solutions fail to overcome two main key issue for us: **context usage** and **task failure**.

The context knowledge is key for task planning. Without a proper knowledge, the planner can not generate an adapted sequence of tasks. Although essential, how the context is acquired is often not addressed in automated planning contributions. Yet, for a robot in a smart environment, how the context is acquired and its cost is not to be neglected. In fact, the context acquisition implies going through several processes, including sensors data gathering, transformation, and reasoning, as seen in previous chapters. These processes can be costly, for example VARSeR is a heavy visual recognition process that uses a lot of resources. Similarly, some devices, such as smart phones, rely on battery and should be preserved. When generating a plan, the planner does not necessarily use all the context data. For example, if Nono decides to go recharge its battery, the knowledge of Arthur's activity is irrelevant when making the plan to go to the charging station. Using a visual process to infer Arthur activity would be a waste of time and energy for the robot. Even if the context is continuously monitored for decision making, once the planning process has started, the context acquisition can be limited to the planning needs. In brief, the planner should not ignore the effect of the context acquisition it requires and should **only use the relevant information**. To the best of our knowledge, such an issue was not addressed in the literature, or at least not in a robotic and/or

smart environment context.

Another issue encountered by domestic robots are the task failures, in particular due to context changes or incomplete knowledge at planning. In fact, as the user lives, the environment varies. When executing a plan, by the time the robot achieves it, context changes may cause the plan to be outdated. For example, Nono may go fetch a glass of water to give to Arthur in the living room, but by the time it gets the water, Arthur has moved to the bedroom and Nono fails to deliver. Similarly, some information are not available at planning, but will be obtained during the plan execution. Typically, if a door is not equipped nor monitored by any sensor, the robot can only have the information of its state, closed or open, when it sees the door. These two issues can cause the plan to be mistaken, thus engendering failures. In the literature, failures are generally handled by creating a new alternative plan [183, 131, 71], even if it said to be a consuming process [185, 56]. Although simple, such a solution is far from being suitable. Firstly, with such failure management, the robot has to endure the failure itself, which can be time-consuming and even hazardous, in a case of a fall for instance. This means that the effort put in the execution of tasks prior to the failure are wasted. For example, if Nono is asked to go to the living room, but then realizes the door is closed once it reaches it, Nono would have waste the time and effort to reach the door already. Furthermore, the time needed to execute the alternative plan also delays the achievement of the goal, which may look unacceptable in the eyes of the user. For these reasons, failures and replanning should be avoided as much as possible. In the literature only a few works addressed the issue. Weser et al. tried to limit the usage of replanning when having incomplete knowledge [185]. They try to avoid replanning and acquire context during execution. Yet, although they observe at runtime, the adaptation to context changes and failure management relies on replanning. Other works tried to adapt the plan, such as RepairSHOP [183], but requires a heavy expert knowledge. Hanheide and al proposed to observe at runtime to understand failure [71], yet, in the optic to plan again. In fact, there are no solutions that are efficiently able to cope with the **dynamism of the environment**.

The uncertainty of the context knowledge is also an issue when planning. Indeed, they can lead to inappropriate task selection, causing failures. However, in this work, we focus on the two aforementioned problems as we believe them to be more critical and should be tackled prior handling uncertainty. Uncertainty in planning is explored through POMDP [83, 14], but this approach lacks flexibility. Generating a plan supporting uncertainty is pointless is sure to fail by not adapting to context changes. On the same level, managing uncertainty can be eased by focusing only on the relevant data. That is why uncertainty handling is an improvement of the presented contribution. Nevertheless, uncertainty was tackled in our previous contributions and enabled an, accurate decision making. Thus, the certainty of the planning goal is ensured.

In this chapter, we present a novel planning approach called Dynamic Hierarchical Task Network: (DHTN). DHTN relies on the formalism of HTN [170]. We selected HTN for its hierarchical approach and the knowledge it carries: HTN ensures the quality and relevance of selected task sequence as they were provided, thus validated, by an expert. DHTN takes advantage of HTN hierarchical structure to improve it over two core aspects:

- **Goal Oriented Observation**: Most planners generate the plan according to the observations, to the sensing. DHTN however relies on a planning to sensing approach. The idea is to define

what context data to use, and consequently what to observe, according to the current goals of the planner. By doing so, the planner only uses the data that matters in its planning, thus minimizing the required volume of data.

- **Dynamic planning**: DHTN relies on a new planning paradigm. In fact, in most cases the plan is first created and then executed independently, DHTN on the other hand incrementally generates the plan by performing alternatively planning and execution. This enables DHTN to take into account the last changes of the environment, thus preventing failure due to context change. Further, while controlling the execution DHTN is able to roll back and directly find alternative on the fly rather than replanning.

Let us now see how we incrementally improved HTN toward DHTN. Before entering DHTN itself, it is important to set some definitions and remind HTN formalism. Following, we first address the goal-oriented observation of HTN. We then show DHTN novel planning paradigm and algorithm. DHTN evaluation are then discussed.

12.2 Preliminaries

12.2.1 HTN Formalism

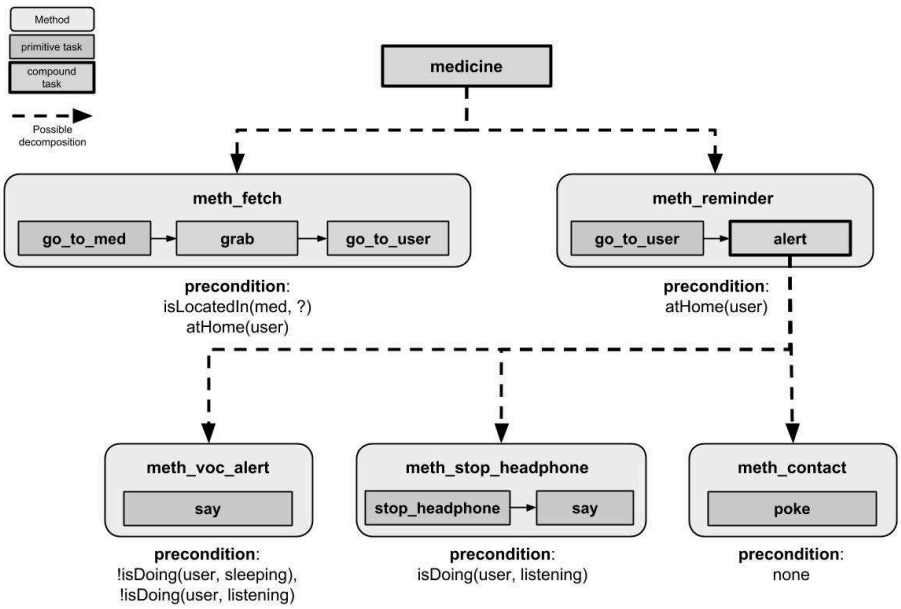


Figure 12.1: Example of a HTN

Simple HTN for a medicine delivery scenario. Two options are possible, either medicines position is known, then the robot fetches and delivers them, or medicines can't be found, then the robot reminds the user. As the user may be sleeping or listening to music, the robot has multiple option to catch his/her attention.

HTN aims to find a solution to a *planning problem* by decomposing tasks into sub-tasks. HTN relies on two types of tasks: *primitive tasks* and *compound tasks*. *Compound tasks* are realized by

sub-tasks, while *primitive tasks* are *ready-to-run* tasks and cannot be decomposed. The result of the planning process is called a *solution*, it is a totally ordered set of *primitive tasks*.

A *planning problem* is defined as a 3-tuple $(g, s, \mathcal{D})$, where g is a sequence of *tasks* to achieve, i.e. a *goal*, s is the initial *state* of the environment defined as predicates, and $\mathcal{D}$ is the *domain*. The *domain* $\mathcal{D}$ is a pair $\mathcal{D} = (\mathcal{O}, \mathcal{M})$, where $\mathcal{O}$ is a set of *operators* and , $\mathcal{M}$ a set of *methods*. An *operator* is a concrete executable action and it solves a *primitive task*. A *method* indicates how to decompose a *compound task* into a partially ordered set of sub-tasks, each of which can be *compound* or *primitive*. It is defined by: its name, the task it 'realizes', preconditions as predicates describing when it is applicable, and a sequence of *primitive* or *compound* tasks, (i.e. the sub-tasks). To represent the world state and preconditions, HTN relies on *predicates*. A *predicate* represents a property of the current context and respects the following pattern: *property(id1, id2, ...)*.

To generate a *solution*, HTN applies *methods* from the *domain* $\mathcal{D}$ to *compound tasks* depth-firstly, starting by goal tasks from g.

We summarize in the following the HTN procedure: first of all, it selects the first *compound task* ct in g; then it selects a decomposition (a *method*) from $\mathcal{D}$, replaces ct by the selected decomposition and repeats until no *compound task* remains. *Methods* are selected thanks to their preconditions and according to the state of the environment. If no *method* is applicable for a *task*, the algorithm backtracks to the *parent task* and tries to find another *method*. The last step consists in applying the *operators* from the *domain* $\mathcal{D}$ to each primitive task of the *solution*.

12.2.2 Definitions

Planning is closely related to the context knowledge and rely on context data. Context data is formally expressed in Definition 10. In case of HTN formalism, these context data are formalised using a predicate form as follows: $predicate(subject, object)$. Note that, a predicate can be seen as a type of data, while valued predicate, i.e. propositions, can be seen as instance of context data. Context data are usually formalized as RDF in previous contributions, they can be translate to a predicate formalism, and vice-versa, with ease.

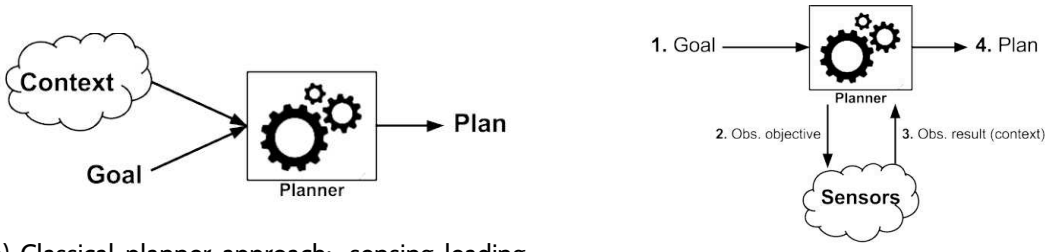
12.3 Goal Oriented Observation

The knowledge of the context is key in task planners. However, in most works, how the context is observed and acquired is not addressed. In fact, it is supposed known. In reality, acquiring context data is an important process that requires gathering from sensors, transformations, filtering and reasoning, as performed in FSCEP or VARSeR. Furthermore, the resulting knowledge can be massive, causing the research space to be large. In classical hierarchical planning, preconditions are checked any time a compound task decomposition is being evaluated. Consequently, every time a method is selected, its preconditions are compared to all the predicates carried in the context knowledge. This process can be refined: the research space can be reduced according to the planning needs. The idea is for the planner to actively pull only the relevant data instead of passively using all the context knowledge. Doing so also enables the possibility of reducing the impact of the planner on context acquisition process as observation can be focused on particular context data.

A first refinement can be performed by reducing the context knowledge to the data that are used in the HTN. In fact, not all context data are used as preconditions. $\mathcal{P}_{htn}$ represents all predicates, hence all context data, that are used in the hierarchy of tasks. It is defined as follows.

Definition 26. *Predicates for a HTN knowledge $\mathcal{P}_{htn}$:*
Let $\mathcal{P}_{htn}$ be the set of all predicates in HTN. Let $\mathcal{M} \in \mathcal{D}$ be the set of methods belonging to the HTN domain. $\mathcal{P}_{htn}$ corresponds to the union of all methods preconditions (i.e., predicates).

$$\forall m \in \mathcal{M}, \mathcal{P}_{htn} = \cup precond(m),$$



(a) Classical planner approach: sensing leading the planning

(b) Novel approach: planning leading the sensing

Figure 12.2: Differences of context acquisition approaches for planning

By considering $\mathcal{P}_{htn}$, the planner uses a more specific set of predicates. However, this can be even more refined by adopting a planning to sensing approach. The idea is to define the set of relevant predicates according to the needs of the planning phase. By doing so, the planner will only use the required and specific context data from the context knowledge. Our proposal consists of enabling *the goal oriented context acquisition*. In fact, as depicted in Figure 12.2, the planner defines what is to be observed, thus reducing even more the research space and the impact on the smart environment. To do so, we define $\mathcal{P}_{goal}$n the set of predicates to acquire for a given goal:

Definition 27. *Predicates for a given goal ($\mathcal{P}_{goal}$)*
Let $\mathcal{P}_{goal}$ be the set of predicates to acquire for a given goal g in an HTN domain $\mathcal{D}$. It is computed by the function $List\_Predicates$ as illustrated in Algorithm 1.

$$\mathcal{P}_{goal} = List\_Predicates(g, \mathcal{D})$$

For example, in the HTN described in Figure 12.1, $\mathcal{P}_{goal}$=`{ isLocatedIn, atHome, isDoing }` . Thus, the planner will only compare predicates to the propositions matching those predicates and won't bother pulling temperatures for example.

By computing and using only the predicates carried in $\mathcal{P}_{goal}$, the planner pulls a specific knowledge. Adoption of such a goal oriented context acquisition, not only reduce the research space when evaluating preconditions, in also reduce the impact of the planner on the robot and the smart environment. In fact, once the decision, through CAREDAS for example, was made, the context acquisition can be limited to the strict needs of the planner. For example, if the user activity is not required to compute the plan, VARSeR can be disabled to prevent waste of time and energy. When HTN is used on its own, a mapping between predicates and sensors can be used. In that case, by using a goal oriented context acquisition, only the relevant sensors are queried, preventing waste usage

of devices. Such an approach allow to be more compliant with a smart environment, however, the dynamism of the surroundings is yet to be tackled, as we discuss in the next section.

Algorithm 1 $\mathcal{P}_{goal}$ computation: $List\_Predicates$

Input: A goal g, a domain $\mathcal{D}$
Output: A set of predicates $\mathcal{P}_{goal}$

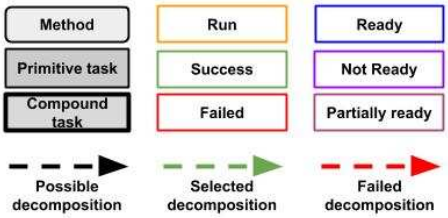
```
1: function LIST_PREDICATES(g, D)
2:     P_goal ← {}
3:     for all task in g do
4:         if IS_COMPOUND(task) then
5:             t_meth_list ← SELECT_MATCHING_METHODS(task, D)
6:             for all m in t_meth_list do
7:                 P_goal.ADD(m.preconditions)
8:                 subconditions ← LIST_PREDICATES(m.subtasks, D)
9:                 P_goal.ADD(subconditions)
10:            end for
11:        end if
12:    end for
13:    return P_goal
14: end function
```

12.4 Dynamic Planning and Execution

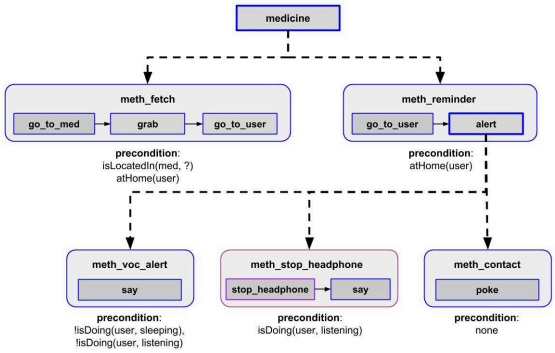
One major issue is remaining to be tackled: the dynamism of the environment. In fact, home environment, the context varies over times as persons live. In consequence, in order to prevent failure, planning solutions not only need to be well aware of the context, but also to adapt to the context changes. That is why DHTN relies on a new planning approach. Instead of generating the whole plan at once, DHTN incrementally selects methods at runtime and observes the context accordingly. The idea is to decompose a compound task only when reached, oppositely to decomposing all the hierarchy at once. This makes it capable of taking into account the latest changes of the environment preventing blocking, useless acting and opportunity missing. This approach is close to reactive HTN [131] that was presented concurrently to DHTN, however this approach lacks context awareness. DHTN planning approach implies three main features:

- Context observation: as DHTN generates and executes the plan in the same process, it also has to observe in the meantime. It is important to define when and what to observe
- Execution Monitoring: DHTN also handles the execution of the plan. This implies a monitoring of the execution. To do so, DHTN relies on *statuses*, similar to Behaviour Tree [111].
- DHTN Algorithm: DHTN uses a different algorithm than HTN. This algorithm uses all previous features and selects sub-plan according to the current execution advancement.

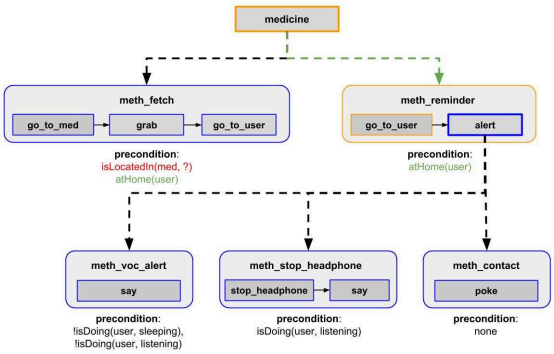
An illustrative example of a DHTN plan management can be found in Figure 12.3. We will now review those three features.



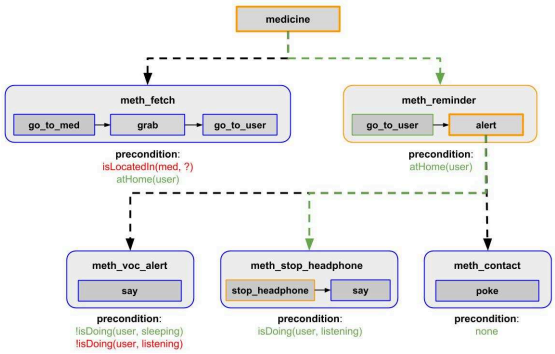
(a) Legend



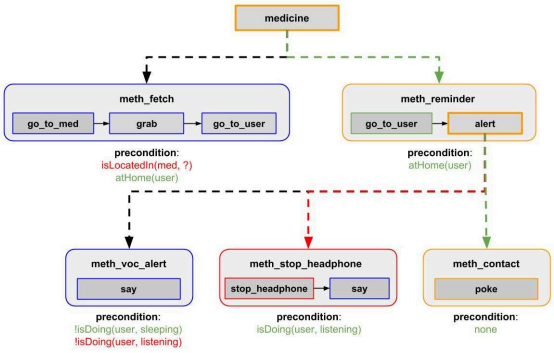
(b) Initial state of the hierarchy



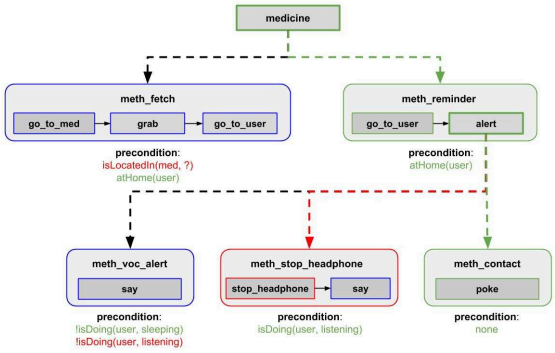
(c) Decomposition of task *medicine*



(d) Decomposition of task *alert*



(e) Failure of task *stop_headphone* and roll-back



(f) Task *medicine* succeeded: end of the plan

Figure 12.3: Example of DHTN decomposition

Example of DHTN planning and execution for medicine delivery scenario.

12.4.1 Context Observation

Like most planning approaches, context awareness is an essential issue of DHTN. But unlike most classical techniques, DHTN relies on a dynamic approach. Hence, defining when and how context observations are made is essential. As we saw in the previous section, HTN can be improved to define the set of predicates that are required in the planning, enabling the planner to pull the relevant context data. Yet, this needs to be refined due to DHTN new paradigm. In fact, in order to determine the set of required predicates for the next decomposition, the observation is not defined according to the general goal, but according to the current upcoming decomposition.

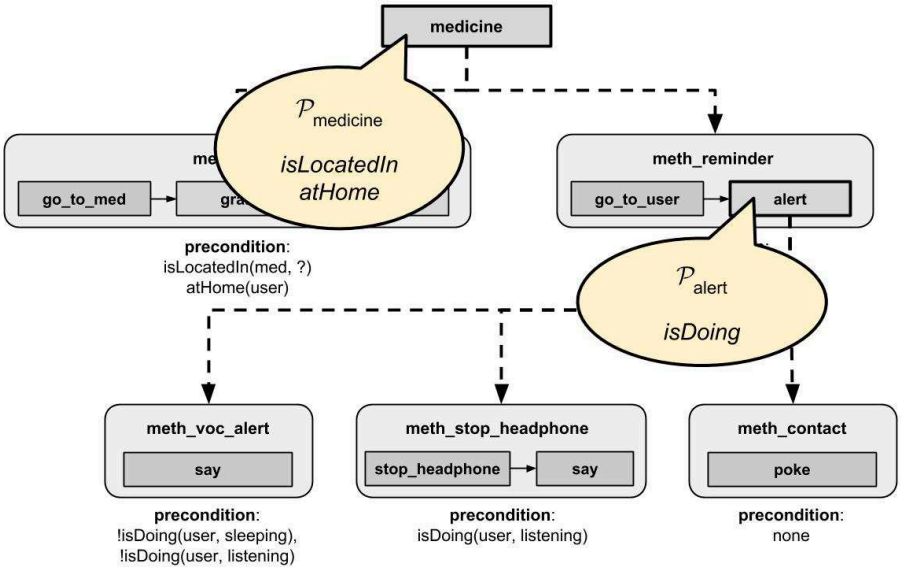


Figure 12.4: Example of resulting observation goals

Whenever DHTN reaches a compound task ct, it needs to decompose it into a method. Accordingly, DHTN will gather context data from the context knowledge. In that case, DHTN does not need to query all $\mathcal{P}_{goal}$, as it only needs some particular predicates to perform the specific decomposition of ct. Thus, when DHTN encounters a compound task ct, it computes the required set of predicates, called $\mathcal{P}_{ct}$. It is computed by gathering all preconditions, including failure causes, of all methods implementing ct. $\mathcal{P}_{ct}$ is defined as follows and its computation process is shown in Algorithm 2.

Definition 28. *Predicates for a compound task ($\mathcal{P}_{ct}$)*

$$\mathcal{P}_{ct} = List\_Predicates\_Task(ct, \mathcal{D})$$

Thus, when decomposing a compound task ct, DHTN computes $\mathcal{P}_{ct}$ and pulls the matching context data from the context knowledge instead of the set of all context data. For example, from the HTN knowledge described in Figure 12.1, two $\mathcal{P}_{ct}$ will be computed: $\mathcal{P}_{medicine}$=`{ isLocatedIn, atHome }` and $\mathcal{P}_{alert}$`{ isDoing }` , as illustrated in Figure 12.4. Every time DHTN reaches a compound task ct, it acquires propositions specifically matching the predicates in $\mathcal{P}_{ct}$. DHTN has then the knowledge to perform the decomposition of ct. By doing so, DHTN only uses specific context data according to the current plan advancement and needs, thus limiting its cost.

Algorithm 2 $\mathcal{P}_{ct}$ computation : $List\_Predicates\_Task$

Input: A compound task ct, a domain $\mathcal{D}$

Output: A set of predicates $\mathcal{P}_{dhtn}$

```
1: function LIST_PREDICATES_TASK(ct, D)
2:     P_ct ← {}
3:     if IS_COMPOUND(ct) then
4:         t_meth_list ← SELECT_MATCHING_METHODS(ct, D)
5:         for all m in t_meth_list do
6:             P_ct.ADD(m.preconditions)
7:         end for
8:     end if
9:     return P_ct
10: end function
```

12.4.2 Execution Monitoring

DHTN not only generates, but also executes the plan. Planning has influence on the execution, and execution has influence on planning. It is necessary for DHTN to be able to monitor the execution. In fact, it has to know what is the current advancement of the plan, which methods were already selected, which methods failed, or if a task can be performed. All these information are critical for planning. For example, Nono can't interact with a turned off heater. Or it should not try to open a door that it already failed to open. In DHTN, this monitoring capabilities are enabled by the usage of **status**. This is directly inspired from Behaviour Tree [111] where nodes and leaves are annotated with statuses that determine the advancement and how the tree is explored. DHTN attaches these statuses to HTN elements and uses them to compute a correct plan, but also to understand the encountered problems. Let us learn more about this notion of status.

In HTN, the domain D is the pair $D = (O, M)$, where O is the set of operators and M the set of methods. In our case, we redefine the domain as $D = \{M \cup O \cup T\}$ where T is the set of primitive or compound tasks. Each element of D is typed: `operator, method, primitive task or compound task`. Intuitively a *status* is an information that characterizes an element of D during its life cycle. Three important phases should be identified: (1) `before execution`, (2) `at execution` and (3) `after execution`. Associated with each phase several values are possible. Each possible status and their related semantic are given in Table 12.5 and explained below. Accordingly, we define this enhanced domain as the Annotated Domain D_a defined as follows and used by DHTN rather than D.

Definition 29. *Annotated Domain* D_a

$\forall e \in D_a$, *there is a status function taking a typed element e of* D_a *at a specific moment and returns a value, called annotations* V, *according to the semantics given in Figure 12.5, such as:*

$Status: D_a$ X $Type$ X $Moment \rightarrow V$

with $Type$ = { `operator, method, primitive task, compound task` },

with $Phase$ = { `before execution, at execution, after execution` },

and V = {`Ready, Not Ready, Partially Ready, Running, Success, Failure, Error, Not Applicable, Unsatisfied`}.

	Before execution	At execution	After execution
Operator	**Ready**: Operator ready to go **Not Ready**: Not operational	**Running**: The operator is running, i.e. the robot is doing an action	**Success**: The operator succeeded **Failure**: The operator failed **Error**: Internal error of the system
Primitive Task	**Ready**: At least one operator is ready to go **Not ready**: No operators are ready for the time being	**Running**	**Success**: Selected operator ended successfully **Not Applicable**: No operators were *ready* when execution was required **Failure**: All operators return *failure* **Error**: Internal error of the system
Method	**Ready**: All subtasks are available **Partially Ready**: Some subtasks are ready **Not Ready**: None of the subtasks is ready	**Running**	**Success**: All subtasks succeed **Failure**: One subtask failed **Error**: Internal error of the system
Compound Task	**Ready**: All method realizing this task are ready **Partially Ready**: Some methods realizing this task are ready **Not Ready**: All methods realizing this task are not ready	**Running**	**Success**: Selected method ended successfully **Unsatisfied**: No method can be selected due to unsatisfied predicates **Not Applicable**: No methods were *ready* when execution was required **Failure**: All methods available failed **Error**: Internal error of the system

Figure 12.5: Status Life Cycle & Semantics

The 'before execution' statuses are simple, yet important. In fact, they indicate if an element is ready to be entered. In its plan, the robot uses its own functionalities, as well as smart devices in the environment. However, devices are not always available, or ready, to be executed. That could be because it is already in use, broken, or simply shut down. As operators are directly related to the devices, or robot functionalities, that perform the actual tasks, statuses of operators directly depend on the availability of devices and/or functionalities. This means that, if the smart heater is busy, the operator in charge of interacting and acting will be marked as `not ready`. We suppose the state of devices and of the robot functionalities are continuously updated, thus continuously updating D_a. The statuses of other elements, that is to say primitive tasks, methods, and compound tasks, are impacted and set in cascade. In fact, as asserted in Table 12.5, statuses of primitive tasks are defined according to related operators, statuses of methods according to compound and primitive task, and

statuses of compound tasks according to methods. If a methods carries both `ready` and `not ready` tasks, its status is set to `partially ready`. As for compound task, their status is set as `partially ready` if there is at least one method `ready` and one `not ready`. Not that, we adopt an optimistic approach, thus `partially ready` statuses count as a ready when setting method and compound task statuses, as seen in Figure 12.3b. For examples, in the example of Figure 12.3, Nono can switch off Arthur's headphone, however, it is busy at first, the related operator is not ready, resulting to the statuses in Figure 12.3b. Yet, by the time Nono reaches the compound task alert, the headphone was reachable and ready, leading to the update of statuses as seen in Figure 12.3c. When planning, if multiple methods have valid preconditions, DHTN will opt for the methods with a `ready` status over methods with a `partially ready` or `not ready` one.

When DHTN reaches an element, it turns its status to `running`. This defines the current advancement of the plan. If an operator is running, this means the robot is currently performing the given task. Consequently, the related primitive task also has the `running` status, as well as the parent method and parent compound task, and so on. This can be seen as the plan advances in example Figure 12.3, particularly in Figures 12.3c, 12.3d and 12.3e.

Once a task has ended, the outcome defines the operators status of the phase 'after execution'. The device, or the robot, provides the result of its action, thus setting the statuses of the operator. For example, if Nono to have the headphone turned off, the operator status will be set as `failure`. Oppositely, if the robot manages to reach Arthur, the operator related to task *go_to_user* is set as `success`, as in Figure 12.3d. Again, this status is cascaded among other elements, as seen in Figure 12.3e, but not only. In fact, DHTN may encounter a failure of planning. For example, DHTN may fail to decompose as no method have satisfied preconditions, or all method are `not ready`. DHTN fails to advance in the plan, yet there was no task failure. This leads to statuses such as `unsatisfied`, `not applicable`. `Errors` can also occur in the planning for purely technical reasons, such an exception. The information provided by these statuses of failure is useful for logging the failures, as the designer can understand and adjust DHTN if necessary. According to the after execution statuses, DHTN keeps advancing or rollbacks to find alternative, as depicted in Figure 12.3e. When the top task achieves a 'after execution', as in Figure 12.3f, the plan execution is over. The outcome of the plan is defined by the status of this task.

These statuses are essential in DHTN algorithm. In fact, they allow it to select a sub-plan whose operators are `ready` and to find an alternative way in case of failure. Typically, when a task fails, DHTN goes back to the mother compound task and tries to pick another one that is `ready`. In conclusion, these statuses allow to control the execution, but also allow to dynamically generate a plan that is ready to be executed and adjusted in case of failure. Let us now see how the algorithm uses statuses.

12.4.3 DHTN Algorithm

As its core, DHTN aims to plan and execute alternatively. By doing so, DHTN is able to take into account the last changes in the environment. Performing planning and execution requires a novel algorithm. DHTN algorithm consists in going through the hierarchy of tasks according to the statuses, executes primitive tasks, and decomposes compound tasks only when reached. By doing so, DHTN executes the plan as it builds it. DHTN recursive algorithm is described in Algorithm 3. Note that the

Algorithm 3 DHTN recursive algorithm

Input: An initial task t, the annotated domain D_a
Output: Status for the task t

```
1: function DHTN_RUN(t, D_a)
2:     if t.status ≠ READY then
3:         t.status ← NOT APPLICABLE
4:         return t.status
5:     end if
6:     t.status ← RUNNING
7:
8:     if IS_COMPOUND(t) then
9:         sel_meth ← null
10:        t_meth_list ← SELECT_MATCHING_METHODS(t, D_a)
11:        w ← GET_CONTEXT_KNOWLEDGE(t)
12:        sel_meth ← SELECT_BEST_METHOD(t_meth_list, w)
13:        if sel_meth = ∅ then
14:            t.status ← UNSATISFIED||NOTAPPLICABLE
15:            return t.status
16:        end if
17:        while sel_meth ≠ null do
18:            for all st in sel_meth.subtasks do
19:                res ← DHTN_RUN(st, D_a)
20:                if res ≠ SUCCESS then
21:                    break
22:                end if
23:            end for
24:            if res = SUCCESS then
25:                sel_meth.status ← SUCCESS
26:                t.status ← SUCCESS
27:                return t.status
28:            end if
29:            sel_meth.status ← FAILURE
30:            w ← QUICK_OBSERVATION(sel_meth.subtasks)
31:            w ← GET_OBSERVATION_RESULT()
32:            sel_meth ← SELECT_BEST_METHOD(t_meth_list, w)
33:        end while
34:        t.status ← FAILURE
35:        return t.status
36:
37:    else if IS_PRIMITIVE(t) then
38:        sel_op ← null
39:        res ← FAILURE
40:        t_op_list ← SELECT_MATCHING_OPERATORS(t, D_a)
41:        sel_op ← SELECT_BEST_OPERATOR(t_op_list)
42:        if sel_op = ∅ then
43:            t.status ← NOTAPPLICABLE
44:            return t.status
45:        end if
46:        while sel_op ≠ null and res ≠ SUCCESS do
47:            res ← RUN(sel_op)
48:            sel_op ← SELECT_BEST_OPERATOR(t_op_list)
49:        end while
50:        if res = SUCCESS then
51:            t.status ← SUCCESS
52:            return t.status
53:        else
54:            t.status ← FAILURE
55:            return t.status
56:        end if
57:    end if
58: end function
```

implementation differs from the described algorithm.

DHTN starts with a task, that is the goal of the planning. For example in Figure 12.3, the goal is to achieve the compound task *medicine*. DHTN ends when this task is affected with a post execution statuses, that is the result of the whole planning, as in Figure 12.3f. Consequently, note that this task will have the status `running` during the whole execution of the plan. DHTN algorithm goes from task to task. When it entered a task, including the starting one, DHTN sets it status to `running` and handles it. DHTN has two distinct ways for handling compound tasks and primitive tasks:

- **Compound task management, i. e. planning** (from line 8 to 35 in Algorithm 3):
  When DHTN deals with a compound task, it aims to decompose it into a method. In other words, DHTN plans the upcoming tasks. This is illustrated in Figure 12.3c and 12.3d. First, it gathers the context knowledge according to the computed goal observation. From this, it selects the best method according to the statuses and preconditions. If a method can't be found, the compound task fails. If a method has been selected, DHTN is **reapplied** on each subtask of this method. Once all subtasks ended, the method is a success, thus the current compound task is success. If a subtask fails, DHTN rolls back and then tries to apply another method, as depicted in Figure 12.3e. If all methods failed, the task failed. Once ended, DHTN goes back in the recursion and continues planning.

- **Primitive task management, i. e. executing** (from line 37 to 56 in Algorithm 3):
  When DHTN handles a primitive task, it aims to execute it. To do so, it first finds a `ready` operator using the statuses. Once found, the operator is executed. The execution is blocking, thus DHTN waits for the task for being physically performed. Once the operation has ended, DHTN retrieves the results and behaves accordingly. If the operation was successful, thus the task is a `success`. If not, DHTN tries to find another operator and repeats the process. If all operators failed, the task fails. Once ended, DHTN goes back in the recursion and continues planning.

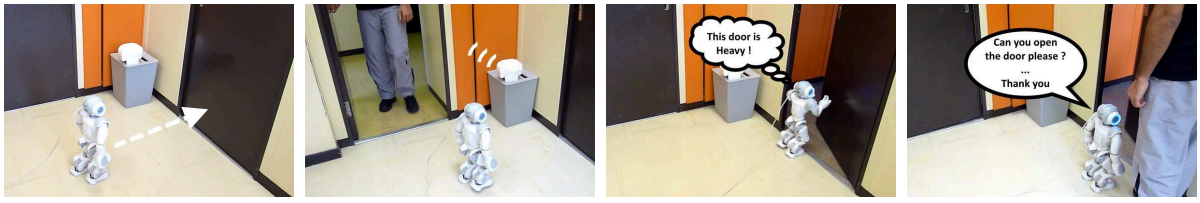
As we can see, DHTN goes through the hierarchy of tasks, task per task, as we show in Figure 12.3. In the case of a compound task, DHTN performs a decomposition, thus planning, with the latest knowledge of the context and statuses of methods. If the task is primitive, DHTN runs an operator, thus physically executes the action, and wait for the robot or any devices to perform the action. According to the outcome of the execution and the advancement of the plan, the planner updates the statuses. Consequently, in the end, DHTN has generated and executed a plan, taking into account the last changes of the environment, and using only the sensors that were relevant. This makes the robot using DHTN highly efficient in smart home environments, as our experiments pointed out.

12.5 Experiments

DHTN was prototyped and experimented to evaluate its ability to adapt to environment changes and to observe according to the goal. Hence, we confronted DHTN to simulated changing context. We then measured the number of replanning required, as well as the number of task executed, to reach a goal. We also measured the usage of smart devices. DHTN was compared to classical planning approach, in that case HTN as it is the closest planner to our work.

12.5.1 Implementation

DHTN was implemented in C++. It was tested on a Nao robot using ROS. Action of Nao were conceived in Python and could interact with our planner through ROS. Appendix B includes further illustrations of interactions between ROS and DHTN. The HadapTIC platform was again used as a



(a) The robot is advancing to the door (b) A human enters the room and is detected (c) Without DHTN: the robot ignores the human (d) With DHTN: the robot asks the human for help

Figure 12.6: Scenario of our test of DHTN with a Nao

smart environment. Figure 12.6 shows pictures of a conducted test were the robot is able to take into account the changes occurring during the execution of the plan.

In order to be evaluated, DHTN was confronted to simulation through Freedomotic. Plugins and devices were added, such as a representation of the robot or plugin to communicate with DHTN. The simulation was able to generate context data, and create variation in the context.

12.5.2 Protocol

DHTN was evaluated on two axis.

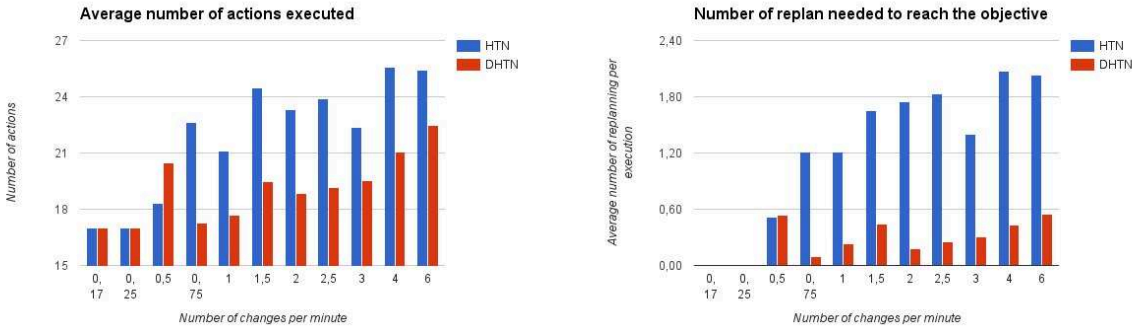
Firstly, we compared HTN and DHTN over the dynamism of the environment. To do so, we considered a scenario implemented in Freedomotic. The scenario consist of Nono having to deliver a glass of water to Arthur. It has to go to the kitchen, and then reach Arthur. However, while the robot, Arthur can go from one room to another and open or close doors, such changes require an adaptation of the plan. We studied the behaviour of planners with multiple levels of dynamism of the environments, represented by the frequency of changes in the context, measured in changes per minutes. This means that, periodically, a random value of the context is changed, for example the user location may change or a door may close or open. This is performed by a custom Freedomotic plugin and context data are carried by virtual sensors and objects. These data are then communicated to DHTN through a ROS interface, that formats the context data as predicate usable by our planner. For each frequency, the scenario was run 50 times for each planning system. We measured the number of actions required to reach the goal, the number of blocking encountered and the number of replanning. Replanning means recalling the planner when it failed, for DHTN it means all options failed and the process had to be restarted. To prevent permanently replanning, a replanning can't be applied if a same action fails two time in a row, in that case, the robot fails to achieve its goal. Please note that it is possible to use other rules to determine the incapacity to reach the goal, but this one is enough for our simulation. Results are shown in Figure 12.7.

Secondly, in order to analyse DHTN's context awareness, we measured the number of sensor query in three different modes:

- Continuous: Data are acquired at a given frequency, independently from the plan.
- At selection: All the context is analysed, but only when a method is going to be selected
- Smart: Only the required devices are queried when needed.

DHTN interacts with those virtual sensors through a ROS interface and UDP communications toward Freedomotic devices. For all three cases, the simulation was executed in the exact same context using our approach, DHTN. Furthermore, no task failure were encountered in these tries. For analysing the context continuously, an update frequency shall be defined. In our case, we think observing once per executing action is enough. In our simulation, continuous observation mode analyse all the context each time a new action is executed. Results are displayed in figure 12.8.

12.5.3 Results



(a) Comparison on the average number of actions required to reach the goal

(b) Comparison on the average number of replanning required to reach the goal

Figure 12.7: Results of the HTN/DHTN comparison

Results of the comparison are presented in figure 12.7. As depicted in sub-figure 12.7a, DHTN requires less actions to reach the goal than HTN. This is due to the fact that DHTN is able to take into account the last changes of the environment and does not 'wait' a blocking to occur to adjust the plan. As depicted in Figure 12.7b, DHTN requires much less replanning to achieve a goal compared to HTN. This leads to the difference of required actions displayed in Figure 12.7a. Such a difference between DHTN and HTN is explained by the fact that DHTN do not react to task failure, but dynamically adapt to the changes of the environment. HTN on the other, react to the task failure due to the changing of context, thus, it requires more than 1,2 replanning on average when there are more than 0.75 changes per minute. This causes the robot to perform a minimum of 21 and up to 25 tasks to reach the goal with HTN. However, with DHTN, the dynamic approach allows to prevent waste of task execution, thus, the robot only have performed a few tasks. Yet, with DHTN, the number of tasks required of tasks increase with the dynamism of the environment, as DHTN as to perform more adaptations and/or rollbacks. For 1 change per 2 minutes, or 0.5 changes per minute, we observe and artefact, as the number of action required by DHTN is unexpectedly high. This is actually due to the rigidity of our experiment: the change always occurred right when the robot is about to reach the user with the water, in such case, DHTN is not able to handle such a recent change. In the end, DHTN proves to be more efficient as it allows to reach the goal with less tasks in dynamic environment, thus being thrifty in term of time and energy for the robot.

As we would expect, by observing only when a decomposition is required, a lesser number of query is required than a continuous observation. In that case, observing only when selecting a

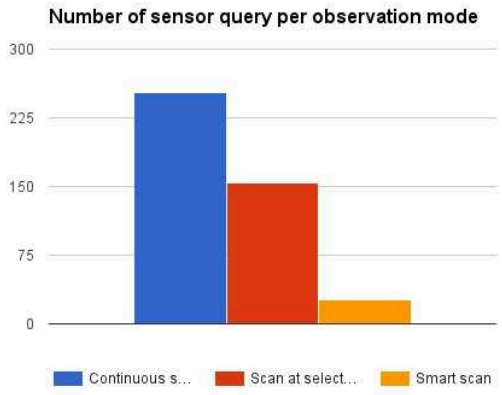


Figure 12.8: Comparison of observation methods

method is 39% more effective than the continuous observation. However, such a figure is quite meaningless, as the difference vary depending on the scenario and the design of DHTN knowledge. Nevertheless, observing when decomposing is naturally more thrifty. The most interesting points concerns the difference between smart observation and observation at select. As depicted, observing smartly, that is to say querying only the implied devices, is 83% more effective than the observation at selection. Of course, such a difference depends on the number of sensors, the variety of type and the selected plan. Still, by observing the context according to the planner needs, we are able to use the environment more efficiently than any other approach.

All in all, through these experiments, DHTN has proven its ability to face dynamic environment. In fact, thanks to its dynamic planning, DHTN enable the robot to reach its goal with a limited number of task compared to using a classical planner. In fact, the instead of reacting to a failure and then adapt, DHTN dynamically adapt to the changes, preventing the execution of task that are already outdated in the plan. Note that, in environment static or with low context changes, DHTN and HTN are similar. Furthermore, by using only the context data it needs, DHTN greatly reduces the number of sensor query, thus having a reasonable usage of the smart devices.

12.6 Conclusion

In this chapter, we presented DHTN, a novel planning approach that executes and generates the plan in a same process while limiting the usage of the smart environment. In order to cope with the constraints of smart home context, DHTN relies on three main features. First, DHTN is able to compute and use a set of predicates that are relevant. By doing so, DHTN does not gather context data that won't be used. This allow to prevent wasting devices usage and can lead to save time and energy in case of heavy process, such as vision based activity recognition. Secondly, DHTN relies on a novel dynamic algorithm. Instead of planning and then executing, DHTN uses the task hierarchy to plan and execute incrementally. This enables it to be flexible facing the changes of the environment, preventing failures due to these variations. It is also able to find an alternative if a failure couldn't be avoided.

DHTN was prototyped and tested through simulation as well as on a real personal robot. Our results show our contribution to enable a reduction of number of tasks to execute in order to reach a goal as it takes into account the last changes of context, thus fails less. Furthermore, thanks to its goal oriented context acquisition, DHTN only uses a limited number of data and, if used as its own, only queries required devices. This proves the relevance of DHTN for robots in smart environment. However, there are multiple axis of improvements. First, proposing a cost model for task in term of time and effort seems to be an essential yet demanding improvement. With such a model, DHTN would be able to generate accurate plan in time, including observation time window, as well as generating plans that require minimal resource and time of execution. Second, but not least, comes the uncertainty of data. In fact, in FAIRIE, uncertainty is tackled until the decision making, enabling an accurate decision, yet, uncertainty of data can have an impact on the planning and causes failure. Thus taking into account uncertainty when evaluating preconditions is an important perspectives. Note that these two axis of improvement can be tackled together, as the cost model could consider the confidence of the selections based on uncertainty.

DHTN is able to prevent failure due to context changes. To do so, it relies on a hierarchical knowledge, including preconditions, provided by an expert. However, in open environments, it is impossible to foresee all possible conditions ensuring a failure proof execution. For example, if Nono does not know Arthur is a hard sleeper and is not woken up by sound alerts, it will fail to alert him from a danger when he sleeps. Such conditions can hardly be specified and need to be learnt. This what we explored through LEAF, addressed in the next chapter.

Chapter 13

From Actions to Experience

13.1 Introduction

Failures are part of people experiences. Yet failing is often perceived as unacceptable and shameful. This is also true for service robots. Unlike industrial robots that operate in dedicated environments, services robots have to perform their tasks in spaces designed around humans, in particular homes. Operating is such environments is challenging as there are numerous sources of problems for robots. Even with the help of smart devices, it is almost impossible to consider and prevent all possible cases in the planning knowledge. In fact, such knowledge is highly dependent of the environment. For example, if not informed, Nono can hardly foresee that Arthur is not awaken by its vocal warning. Overcoming failures is a common problem in personal robotics. As we saw in our state-of-the-art analysis, most works tackle this issue by reacting to it. In those cases, whenever the robot fails, it tries to find an alternative solution by generating a new plan. However, doing so is energy and time consuming, on top of the cost of the failure itself. Indeed, by failing, the robot wastes time and energy in a solution that happens to be pointless. Furthermore, as for humans, failing on a regular basis is not well accepted, hence, a service robot that keeps failing may make the user unsatisfied. In fact, even if failing from time to time is understandable and acceptable, the number of failures should be minimized. Yet, similarly to human, failures are great opportunities for robots to learn. In fact, by using this experience, the robot would be able to understand why it failed and to enrich its planning knowledge. Consequently, thanks to this upgraded knowledge, the robot can proactively avoid failures instead of reacting to it, which makes it quicker and more efficient to reach its goal.

Although it seems to be an essential and promising way, learning from failures was not deeply explored in the literature. In fact, to the best of our knowledge, the works conducted by Sariel and Kapotoglu [157, 86] are the main ones in this field. Let us remember the main features of this approach. As implied, the objective of this work is to enable robots to determine failure causes using previous failed situations in order to avoid using tasks that are expected to fail. The method relies on Inductive Logic Programming (ILP), an experiential learning framework that builds an experience by deriving hypotheses from failure situations. The first step of the process is to create and feed the experience by storing context observations and linking them to a label, success or failure, matching the outcome of the occurring task. Then, with this knowledge, ILP hypotheses are adjusted and

associated with a probability. The higher the probability is, the less ambiguous the hypothesis is. Finally, a POMDP planner was altered to use these hypotheses in the planning process. This allows it to avoid encountering failure by not selecting tasks whose hypotheses of success are negative. In the end, this technique improves the efficiency of planning for the object manipulation case study. However, this approach suffers multiple limitations. Firstly, it relies on a very simple context model and hypotheses. It handles simple case with low complexity and can't cope with high level context data and complex failure causes. Typically, user's activity can be the cause of a failure, for example Nono may fail to warn Arthur while he is sleeping. This requires the robot to be able to reason in order to infer such data, including user activity. Secondly, in this approach, according to the acquired experience, resulting hypotheses may be biased. In fact, by encountering redundant context data that are not related to failures, the solution can identify wrong causes, leading to false positive avoidances. Similarly, causes may be grouped inappropriately. For instance, a pair of context data can be believed as a cause, while only one explains the cause, leading again to false positive avoidances. All in all, the solution proposed by Sariel and Kapotoglu sets the base for using experience to learn and avoid failures, but it functions in simple cases and can't be applied as such for domestic robots in smart environments.

For these reasons, we proposed a novel and more complete solution to learn and avoid failures, called Learning, Evaluating and Avoiding Failures (LEAF). LEAF allows to cope with the requirements of preventing failure situations by avoiding failures due to the holes in the planning knowledge. From a global point of view, LEAF aims to identify causes, to evaluate their relevance, and to allow the planner to adapt to avoid tasks that are likely to fail. It has at its core the following features:

- Creation and maintenance of a semantic ontological-based history. Allowing to cope with highly semantic context data and reasoning. From a robotic point of view, this is a contribution on its own.

- Extraction of cause by considering every candidate independently. This enables the prevention of inappropriate causes association.

- Validation by user to ensure the quality of the learning. It shares similarity with other works, including DCON [10].

- Modelling of the approach as a multi-armed bandit problem[1] and usage of multi-armed bandit solutions. It aims to make the cause extraction and validation efficient according to the context and current knowledge of causes.

- Improvement of the multi-armed bandit approach with causal graph computed from causal induction applied on the robot experience. By doing so, we target a more accurate cause extraction.

With these features, LEAF is able to overcome the current limits of the state-of-the-art and allows to achieve our requirement of failure handling. In this chapter, we detail LEAF through four

[1] https://en.wikipedia.org/wiki/Multi-armed_bandit

sections. First, we describe the general architecture and principle of LEAF. From there, in the following section, we show how the model can be seen as a multi-armed bandit problem. Then, we describe how we used causal induction and causal graph to improve our multi-armed bandit solution. Finally, we discuss our experiments.

13.2 User Validated Failure Cause Identification

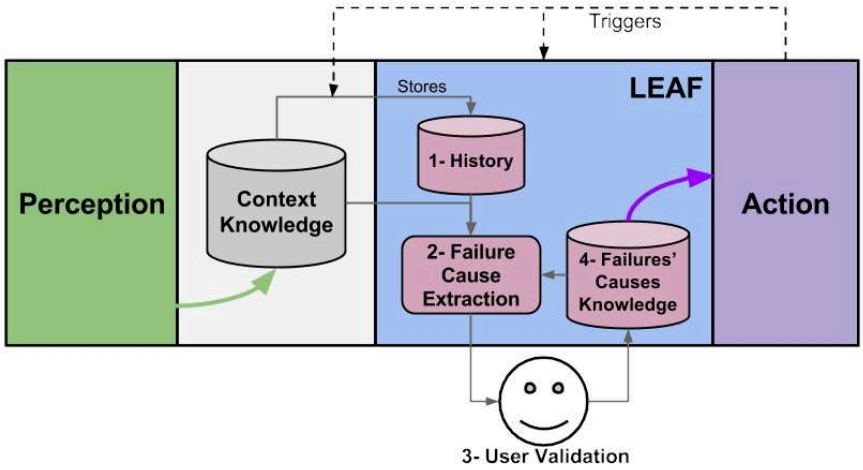


Figure 13.1: LEAF general architecture

LEAF aims to identify causes of tasks failure by taking into consideration two aspects: a high level knowledge to cope with possibly complex causes, and a high accuracy of identification to efficiently avoid further failures. To achieve so, LEAF relies on a four steps process, described in Figure 13.1:

1. First, LEAF acquires and stores the experience of the robot through an **History**. The history stores all situations encountered by the robot when it performed a task and when it failed or succeed. In fact, whenever a task end, the current situation carried in the context knowledge, referred as live situation, is stored in the history and associated with the performed task and its outcome.

2. Then, whenever a task fails, the robot tries to understand what can explain it. LEAF will **extract possible failure causes from the experience**. To do so, it analyses the history, the live situation, and the current knowledge of previous acquired failure causes. It extracts a list of possible causes that can then be submitted to checking.

3. Thirdly, the **user** is asked about the extracted failure cause to **confirm the cause identification**. The user can be asked by several manner, including a vocal question such as: *"Did i fail to do this task because of that?"*. According to the user's answers, the causes are retained or set aside, and are associated with a trust weight.

4. Lastly, the retained causes are stored, and/or updated, in the **failure cause knowledge**, that carries known possible failure causes for a task. This knowledge is then used by the planner

that will avoid selecting task whose failure causes are observed.

Note that, as implied, the process is not synchronous, meaning that it is not executed completely at once. In fact, the history is acquired whenever a task is completed by the robot while the cause extraction is triggered only when a failure occurs. Similarly, user validation is not necessarily performed right after the cause extraction: the extracted causes are stored and the user is asked whenever he/she is available. In the meantime, other failures can occur and other situations will be stored in the history. Let us now review more specifically each step in the next subsections, before discussing how it can be seen as a multi-armed bandit problem in the next section.

13.2.1 Situation History

The **History** represents the experience of the robots. It can be seen as snapshots of situations previously encountered. Each situation in the history is associated with a task and its outcome. With such knowledge, the robot, through LEAF, know what happened when it failed or succeeded. The history is created by storing the current situation stored in the context knowledge.

The context knowledge is maintained through the perception layer, in other words, our previously presented contributions can be used, as well as other tools. This context knowledge is continuously maintained and updated and represents the current situation or **live situation**. The live situation is a set of context data that are currently observed. Again, we rely on Definition 10 for context data.

Similarly as previous contributions, this context knowledge is maintained as an ontology where each context data is represented as a RDF triple. This enables the application of rules for inferring high level context data. Furthermore, it carries outcome of previous modules, including recognized user activities. For example *(arthur isDoing music)* is a context data representing the activity of Arthur of listening to music.

As expressed in Definition 20, a situation is a set of context data. Whenever a task ends, the live situation is "snapshot" into the history. However, Definition 20 of situations does not take into consideration the outcome of tasks. Hence, LEAF uses a refined definition of a situation:

Definition 30. *Situation (ST):*
A situation is a set of CD that have occurred at a given time interval. In this work, we are interested in the situation during one robot's task. A situation can be seen as a 'snapshot' of the state of the environment during one task. Formally, we define a situation as a 5-tuple $(\{CD_i\}, status, task, t_s, t_e)$. $\{CD_i\}$ *is a set of context data captured during a time interval* $[t_s, t_e]$*;* $task$ *is the task the robot was doing during this situation;* $status$ *is the outcome of the execution of the* $task$*, it is set to "null" when the situation is created and is set either to "success" or "failure" after the task execution;* t_s *is the start time of the execution of the* $task$ *and* t_e *is the end time of its execution. In the rest of the paper, we refer to situations as either failure situation or success situation when their status are respectively equal to failure or success.*

Thus, the history is a set of situations defined as follows:

Definition 31. *History (H):*
H *is a set of situations* ST*. We denote* H_t *the set of situations* $ST \in H$ *where* $task = t$*.*

Consequently, the history H also relies on an ontological format. Thus, in terms of representation, it can be seen as a set of ontologies, that are situations ST. Be aware that the implementation may differ from this design for optimization reasons. Figure 13.1 shows an example of a subset H_{vocal_alert} of H. By storing a high number of past experience and using a rich context model, namely ontologies, the history of LEAF enables to **detect possibly complex failure causes**, such as the user activity. Oppositely to a simple data model used in state-of-the-art approaches. Consequently, this knowledge is used to identify possible causes of task failure.

Sit.	Status	Start	End	Observed context data (**failure causes are bold**, *inferred data are italic*)
S_1	Failure	18/03/17 18:30:50	18/03/17 18:40:55	(arthur isLocatedIn livingroom), **(arthur isDoing music)**, (livingroom hasSoundLevel 5db), (nono hasVolumeLevel 30db)
S_2	Failure	18/03/17 20:10:30	18/03/17 20:15:30	(arthur isLocatedIn livingroom), (arthur isDoing tv), (livingroom hasSoundLevel 75db), (nono hasVolumeLevel 30db), ***(arthur vocUnreachTo nono)***
S_3	Success	19/03/17 10:30:50	19/03/17 10:34:15	(arthur isLocatedIn livingroom), (arthur isDoing tv), (livingroom hasSoundLevel 40db), (nono hasVolumeLevel 30db)
S_4	Failure	19/03/17 11:00:50	19/03/17 11:03:55	(arthur isLocatedIn livingroom), **(arthur isDoing music)**, (livingroom hasSoundLevel 20db), (nono hasVolumeLevel 30db)
S_5	Success	19/03/17 15:06:35	19/03/17 15:08:55	(arthur isLocatedIn bedroom), (arthur isDoing reading), (livingroom hasSoundLevel 25db), (nono hasVolumeLevel 30db)
S_6	Failure	20/03/17 10:15:50	20/03/17 10:18:00	(arthur isLocatedIn livingroom), (arthur isDoing phoning), (livingroom hasSoundLevel 65db), (nono hasVolumeLevel 30db), ***(arthur vocUnreachTo nono)***

Table 13.1: Example an experience history

Example of a H_{vocal_alert}, a history for a task task 'vocal alert' that fails due to excessive noise or user activity. Real failure causes are bold.

13.2.2 Failure Cause Extraction

The cause extraction step uses the history to identify possible causes of failure. A failure cause is defined as follows:

Definition 32. *Cause (C):*
A cause is a CD that fully or partially explains a failure situation. Formally, we define a cause as a couple (CD, ST) where $task = t$ and $CD \in ST$ is the context data that causes the failure of the situation ST.

For example, in the failing situation: (*{(arthur, isLocatedIn, kitchen, 15:01:10), (arthur, isDoing, music, 15:3:44)}*, Failure, Alert, 15:00:52, 15:03:00). The failure is caused by the 'music' activity of the user. Note that a failure can have multiple causes. We denote all the causes of one task t as C_t.

The cause extraction process is triggered whenever a task fails. It uses the history of the given task, the live situation, and the failure cause knowledge to extract possible failure cause among the observed context data in the live situation. There are two possible cases: either the identified cause was never encountered, or it was previously identified as a failure cause. The first case consists of "discovering" new possible causes to enrich the cause knowledge. It is mainly used when LEAF has

a small knowledge about failure causes, but is essential to have a complete knowledge of failures causes. The second case principle is to "consolidate" the cause knowledge. In fact, an already identified cause should not be a cause again, as the task would have been avoided otherwise, thus the cause knowledge may be inaccurate. If an already found cause is observed again in a failure situations, it should be checked as the knowledge might be mistaken or biased. Such errors can occur due to variations in the environment, of habit, or simply because the user was mistaken.

We can assert the cause extraction process to be the core of LEAF. It is however strongly dependent of the multi-armed bandit approach. Consequently, we discuss this process after this section. On top of that, causal induction was also used to improve the extraction, particularly for discovering new causes.

Once possible causes are extracted, they are submitted to user validation.

13.2.3 User Validation

The user has supposedly a good knowledge of the environment and can provide accurate information to the robot. To ensure the quality of the cause identification, the user is included in LEAF process to check and evaluate the relevance of extracted causes. This ensures the **quality of the identified failure causes**.

Whenever it has the opportunity, the robot can ask the user about the failure causes it has identified. To do so, the robot can interact with the user through various interfaces, such as a screen display or a vocal interaction. How the interaction occurs is out of the scope of this work. In our experiments, we used the speech capabilities of Nao [63] to generate a vocal exchange. In the interaction, the robot asks the user confirmation for the cause extracted by LEAF. However, only a limited number of "questions" are asked: the robot should ask two or three precise confirmations. In fact, the user should not be the one doing the job, but only confirms some identified causes. In previous example where *(arthur, isDoing, music, 15:3:44)* is the failing cause and identified in the previous process, the question could take the form of: "I failed in task 'alert' earlier: is 'arthur' 'isDoing' 'music' explains why I failed ? ".

From these interrogations, the user can provide multiple answers. We consider a range of five possibilities: {'Yes', 'Probably', 'Partially', 'Possibly', 'No' }. According to this answer, the cause is associated to a weight and inserted in the cause knowledge. In case of an already identified cause, the weight is updated by taking into consideration the previous knowledge.

13.2.4 Cause Knowledge

The cause knowledge carries all the identified failure causes with weights, as well as the tasks that failed due to these causes. These weights, referred to as causality belief (CB), are computed based on user feedbacks, as we will see later.

This knowledge is then used to prevent further failures. To do so, a task planner should take into account this information. The principle is that, if one or multiple failure causes for a task are observed, then the task should be avoided in the plan. To have a short preview, the confidence of the execution of a task is computed by using a logistic differential function based on the weights of

observed failure causes. This metric allow to classify the branches and select the most trustful one. The logistic differential function allows to have a $[0, 1[$ output with an asymptote at 1: the more failure causes are observed, the more the function output is close to, and the lower the planning branch is reliable. By doing so, the task planner creates a plan with limited risk of task failures. The robot has learnt from its experience and enriched its knowledge and its planner: LEAF has then reached its role.

13.2.5 All in All

By having a rich and highly semantic experience and by using user feedback, LEAF is able to identify accurately possibly complex failure causes. But how the cause are extracted? How the user feedbacks are taken into account? In the next section, we will see how LEAF process can be seen as a multi-armed bandit problem. Then, we show how we used a multi-armed bandit solution for cause extraction and user validation.

13.3 A Multi-Armed Bandit Problem

The depicted process is relatively complex and includes various parameters to take into consideration. Hopefully, it can be seen, and/or adapted, as a **multi-armed bandit problem** [110]. This is originally the problem of establishing the gain by using bandits whose expected gain are unknown. Multi-armed bandit solvers aim to maximize reward by efficiently choosing between **exploration**, i.e. using resources to explore new possibilities of gain, or **exploitation**, i.e. ensure gain by using resources from reliable sources. Using a bandit has a **cost** in resources, but provides a **reward**. It is actually a reinforcement learning technique used in various context. We can do the following analogy with our problem:

- Bandit $\Longleftrightarrow$ Context Data (possible cause)
- Exploitation $\Longleftrightarrow$ Consolidation
- Exploration $\Longleftrightarrow$ Discovery
- Cost $\Longleftrightarrow$ User's validation
- Reward $\Longleftrightarrow$ User's feedback

In fact, we aim to select context data that are possible causes to be checked by the user. As previously mentioned, we have two possible cases: either the cause is novel, or it has been already checked. These two cases match the exploitation and exploration notion of multi-armed bandit problem. And finally, as only a limited number of cause can be validated by the user, each "question" costs a "try". User feedback corresponds to the reward: a good cause extraction means a good reward. From this observation, we decided to used a multi-armed bandit technique in LEAF. The technique will define the cause extraction as well as user feedback computation.

There exists multiple multi-armed bandit techniques [27]. Each having their usage strengths and weaknesses. In this work, we are using a variation of R-UCB [27, 28], that is an improvement

of the Upper Confidence Bound (UCB) algorithm [54]. UCB is a well known solution for tackling multi-armed bandit problems. It allows to select the bandit with the higher upper confidence bound when exploiting. This measure enables a selection of the bandit according to previous observations. In UCB, the selection between exploration and exploitation is performed randomly by following a fixed rate. R-UCB improves the UCB by adapting the exploration/exploitation rate according to the current 'risk'. In R-UCB, the selection rate ϵ is dynamic and depends on the risk: the higher the risk is, the less the exploration is performed. R-UCB allows to balance and exploit according to the context, for these reasons, we selected this approach. Among multi-armed bandit techniques, R-UCB appears to be the only one using the context to that extent. Note that we do not aim to do a detailed comparison of existing approaches here, please refer to dedicated work [27] for this matter. Nevertheless, R-UCB isn't perfectly suitable for LEAF and need some adaptations. Let us see how causes are extracted and user feedbacks are taken into account in LEAF.

13.3.1 Enriching or Consolidating the Knowledge

When a task fails, LEAF triggers its cause extraction process that selects, in the live situation, context data that can explain the encountered task failure. There are two types of context data. On the one hand, there are context data that are in the cause knowledge, meaning they were already identified as a cause. On the other hand, there are context data that were not identified as a cause or not met before. Selecting each of them has a different role. Picking a context data that is not known as a cause, or exploring, helps enriching the cause knowledge. Selecting one already identified as a failure, or exploiting, allows to consolidate the cause knowledge. Thus, when does LEAF need opt for one or the other option?

R-UCB balances exploration and exploitation according to a 'risk' variable. In our work, we rather use the notion of *reliability*: the more the robot has previously failed a task, the lower the reliability of the task. This notion allows to orient LEAF toward exploration or exploitation. In fact, a task with a low reliability implies that the cause knowledge of this task is not accurate, as otherwise, the failure should have been prevented. In that case, consolidation, thus exploitation, should be prioritized. Reliability is computed as expressed in Equation 13.1 :

$$R = nbrSucc_N/N \tag{13.1}$$

Where $nbrSucc_N$ is the number of successful situations in the past N situations in H_t. Thus, R is the reliability that represents the success rate of a task t over the past N situations. For instance, for $N = 4$, with the H_t presented in Table 13.1, thus considering S_6, S_5, S_4 and S_3, we obtain $R = 2/4 = 0.5$.

To balance exploration and exploitation, R-UCB relies on the exploration rate ϵ. ϵ is computed as follows in 13.2:

$$\epsilon = \epsilon_{max} - (1 - R) * (\epsilon_{max} - \epsilon_{min}) \tag{13.2}$$

LEAF exploits and explores according to this rate. For example, if $\epsilon = 0.66$, out of three causes to extract, LEAF will extract two possible causes from exploration and one for exploitation. As explained, ϵ varies according to the History and, consequently, R. By doing so, LEAF efficiently balances exploration and exploitation according to the current needs. Note that, in the case where all

observed context data were already checked or none of them were checked, LEAF will respectively only exploit or only explore without taking ϵ into consideration.

Let us now see how LEAF, thanks to R-UCB, extracts possible causes when exploiting and exploring.

Exploitation

When exploiting, LEAF extracts a context data currently observed in the live situation and represented in the cause knowledge. To select the one among all possible candidates, LEAF, through R-UCB algorithm, picks the context data with the highest upper confidence bound d_{cd}. d_{cd} represents the confidence in the selection of cd according to its current causality belief CB, its occurrence and the number of feedbacks already provided by the user. A cd with high causality belief, but few validations, is likely to be wrongly estimated, thus it will have a high d_{cd}, meaning cd needs to be check in priority. d_{cd} is computed as follows:

$$d_{cd} = CB_{cd} * \sqrt{log(F_t)/N_{cd}} \tag{13.3}$$

Where CB_{cd} is the current causality belief of context data cd (see next subsection for belief computation), F is the number of failure situations for the current task t, and N_{cd} is the number of feedback provided by the user for causality of context data cd (for task t).

For instance, let us consider that the robot is in the situation S=*(arthur isLocatedIn livingroom)*, *(arthur isDoing music)* with the history H_t presented in Table 13.1 and cd=*(arthur isLocatedIn livingroom)*; let us assume the user provided feedbacks eight times for cd, $N_{cd} = 8$, and that the resulting causality belief is $CB_{cd} = 0.75$, in that case: $d_{cd} = 0.75 * \sqrt{log(4)/8} = 0.21$. When LEAF exploits, it computes d_{cd} for all observed context data that are also in the cause knowledge. The context data cd with the highest d_{cd} are the best candidates to be checked by the user. This allows to select the cause that is likely to be mistaken.

Exploration

For exploration, R-UCB relies on random selection. It means that LEAF would randomly select a context data that was never selected to be extracted. Random selection is relatively accurate, as it is not sensible to bias. However, with the history, LEAF has a strong experience knowledge that could be used for a refined detection. In fact, random selection can be inefficient when the volume of context data is large while real causes are few. This particularly the case when the robot has no experience yet. Thus, instead of using the R-UCB random exploration, LEAF relies on causal induction to generate a causal graph to identify direct and indirect failure causes. This is actually a consequent improvement of R-UCB and, as such, is presented in the next section.

13.3.2 User Feedback

Once LEAF has determined context data from the cause extraction step, it requests user validation. As mentioned earlier, the robot interacts and asks him about the causes.

We remind that we consider five different user's answers ordered by confidence: {'Yes', 'Probably', 'Partially', 'Possibly', 'No' }. Each is respectively associated to a belief value: { 1.0, 0.75, 0.5, 0.25, 0.0}. This corresponds to the reward, from multi-armed bandit point of view. Based on these feedbacks, the checked context data is associated to a **causality belief** CB and stored in a knowledge base. This causality belief is set as follows: Let B be the belief value from the user's answer. If the context data is new and does not have causality belief CB_{cd} yet, its causality belief is set to B. If the context data was already identified as a cause and has a causality belief, the latter is updated using Equation 13.4:

$$newCB = (oldCB * N + B)/(N + 1) \tag{13.4}$$

Where $newCB$ is the adapted causality belief of the context data and N is the number of previous user answers for this context data (for task t). Let us suppose that the robot asks the user for validation about the context data cd_a*=(katleen isDoing music)* after the failure of the situation S_4. The user answers 'Probably', cd_a was not identified as a cause yet, thus $CB_{cd_a} = 0.75$. Now, let us suppose that the robot encounters a similar situation some time later. However, this time the user is more confident and answers 'Yes'. Hence, the causality belief of cd_a is updated as $CB_{cd_a} = (0.75 * 1 + 1)/(1 + 1) = 0.875$.

Once the user has answered and CB_{cd} are computed, the cause knowledge is updated. This new information will then be used back in the extraction process, but also by the task planner to avoid task that are likely to fail.

13.4 Failure Proof Planning

With the cause knowledge acquired and updated after user validation, it can be used to avoid further failures. This requires an alteration of the planning solution. Typically, if a cause of failure for a task is observed, this task should be avoided in plans. Although various planner could be adapted to use the cause knowledge, as already done with POMDP [157, 86], in our work, we relied and improved HTN. In fact, the cause knowledge is used to enrich the knowledge of HTN by adding a new type of conditions. Consequently, the planning not only select relevant tasks in the context, but also tasks that have low chances of failure. By adapting HTN, the improvement is by extension compatible with DHTN, and more generally, with other approaches using HTN formalism [98, 116]. Note that we rely on HTN notions in this section, please refer to Section 12.2.1 for a presentation of HTN formalism. Let us see how the cause knowledge can be used with HTN.

The general objective is to avoid a task t in the plan if failure causes for t are being observed. To do so, the planning process is altered: when HTN decomposes, it not only checks the preconditions, but also the failing risk of subtasks. When a method has to be selected for decomposing a compound task ct in the planning phase, the following process is executed. We denote $\mathcal{M}_{ct}$ the set of all method realizing ct.

1. Failure causes of all subtasks of $m \in \mathcal{M}_{ct}$ are gathered into one set C_m
2. For each $m \in \mathcal{M}_{ct}$, preconditions are checked
3. For each $m \in \mathcal{M}_{ct}$, with valid preconditions, DHTN computes the risk of failure of m

4. The method $m \in \mathcal{M}_{ct}$ with valid preconditions and the lowest risk of failure is selected.

Let us detail those steps.

The first step of the process is to merge failure causes C_{t_i} of all subtasks t_i of method m. The merging is quite simple: all the identified causes of subtasks $t_i \in m$ are extracted from the cause knowledge C_{t_i} and put in a set C_m that carries all failure causes of the method m. In the case a context data is a failure cause for multiple tasks in m, the maximum CB is selected. Please note that compound sub-tasks are not considered in the computation of C_m: they are tackled when the planning algorithm decomposes them. For example, regarding the task 'alert', the robot has two methods: m_1 =go to user, vocal alert and m_2 =phone alert. Let us consider that LEAF identified two causes for task 'vocal alert': cd_a=*(arthur isDoing listeningMusic)* and cd_b=*(arthur vocUnreachTo nono)*. Thus, in that case C_{m_1} contains cd_a and cd_b.

Once C_m are generated for each possible method m in $\mathcal{M}_{ct}$, methods conditions are checked. Methods whom conditions are not verified are excluded, the remaining set is labelled $\mathcal{M}^*_{ct}$.

The remaining methods failure causes are then checked. The idea is to check if the method would succeed in the current context. Let W[2] be the current context. And let C_{obs} the set of currently observed causes $C_{obs} = W \cap C_m$. Let A be a constant. For each method $m \in M^*_{ct}$, a confidence value is computed through a following logistic differential presented in Equation 13.5:

$$conf_m = 1 - \frac{1}{1 + e^{-ln(A * \sum_{cd \in C_{obs}} CB_{cd})}} \tag{13.5}$$

Equation 13.5 allows to compute a $[0, 1[$ confidence value based on any number of causality belief. With a sum of causality belief of 0, the function returns 1, meaning the task can be safely executed. The confidence then quickly decreases as the sum of causality belief increases and has an asymptote at 0. Thanks to this function, one cause with a high causality belief is enough to engender a low confidence, thus preventing execution. The higher causality is, the lower the confidence, this enables the comparison of tasks confidences. For example, let us consider two causes cd_a=*(arthur isDoing listeningMusic)* and cd_b=*(arthur vocUnreachTo nono)* with $CB_{cd_a} = 0.875$ and $CB_{cd_b} = 1.0$. If the robot is trying to generate a plan in situation S_1=*{(arthur isLocatedIn livingroom), (arthur isDoing music)}*, for $A = 0.75$ confidence value of m_1 is: $conf_{m_1} = 1 - 1/(1 + e^{-ln(1*0.875)}) = 0.6$. In situation S_2=*{(arthur isLocatedIn livingroom), (arthur isDoing music), (arthur vocUnreachTo nono) }*, where two causes are observed, thus the task is more unlikely to succeed, for $A = 0.75$ confidence value of m_1 is: $conf_{m_1} = 1 - 1/(1 + e^{-ln(1*(0.875+1.0))}) = 0.41$.

Having the confidence values for each computed method in M^*_{ct}, the method with the highest confidence value is selected for decomposition. Within the standard HTN, the first method matching the conditions would have been selected. However, a final checking is done before decomposing. In fact, if the confidence value is below 25%, meaning there is 75% of chance the robot won't succeed, the decomposition is aborted. As all other methods are less reliable, the compound task ct is marked to be non-executable. Thus, HTN will backtrack and try to find another option. If a method is successfully selected, HTN can continue its process. In the end, a plan is generated and HTN has checked the possibilities of failures, minimizing the risk of encountering a failing situation.

[2] W stands for world

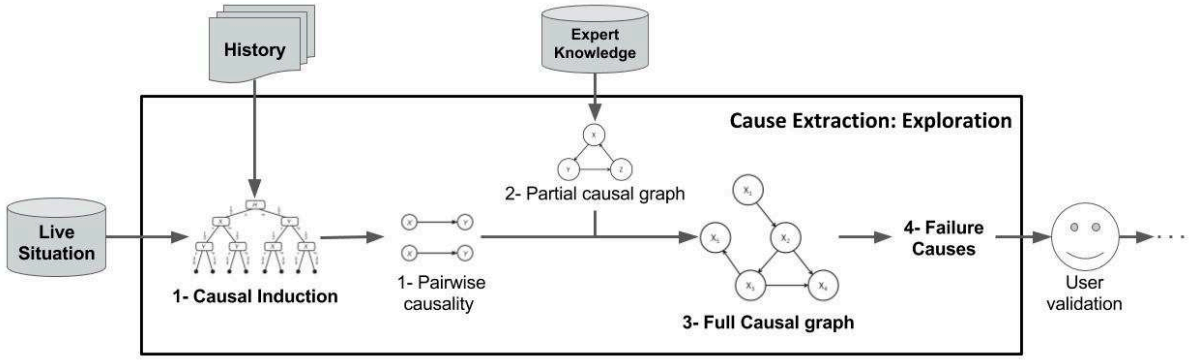


Figure 13.2: Improved exploration process

As defined using HTN formalism, the failure avoidance can be directly applied in our task planner, DHTN. Instead decomposing of compound tasks at once, DHTN dynamically executes and decomposes on the fly, thus computing $conf_m$ at the last moment. As DHTN also monitors the plan, a new status was added on methods: `untrustful`, that is affected if a method as a confidence below 25%. Computing the confidence also impacts context acquisition. Indeed, in goal oriented observation, possible causes of task failure must also be observed. In consequence, the domain is updated as follows: for each $m \in \mathcal{M}_{ct}$, the C_m are added to precondition of m. By doing so, any time $\mathcal{P}_{goal}$ is computed, the causes predicates will be extracted, similarly as other predicates, by the function $List\_Predicates$.

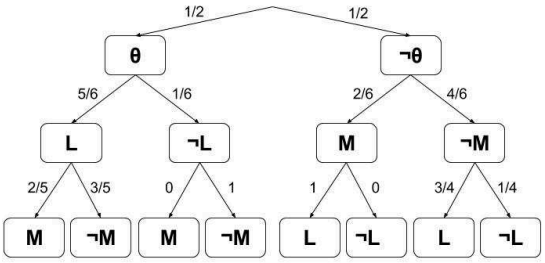
13.5 Causal Induction for Exploration

As mentioned earlier, R-UCB explores by randomly selecting a bandit, that is to say a context data to check in our case. Although it is a decent way of selecting a possible cause, it is limited and can be problematic when considering larger volume of data.

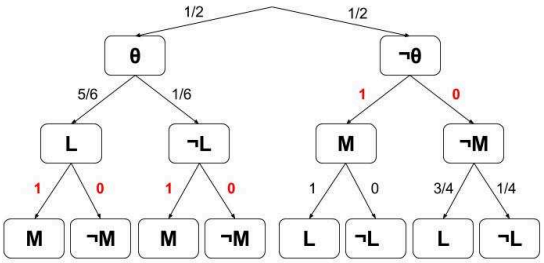
Thus we propose to use the experience acquired by the robot to perform exploration. To do so, we improve R-UCB with causal graph [134] generated from the history through causal induction [130]. In the literature, some works explored the possibility of using causality in multi-armed bandit approaches. Among the most related works, Lattimore et al. [100] address the idea of causal bandits, but they only focus on "pure exploration", meaning they do not consider the exploitation phase: the problem they address is actually different. More closely, some researchers [159, 18] also use causal graph for contextual bandits, however they aim to tackle the issue of co-founders, which is not what we aim to tackle. Our improvement of R-UCB aims to use causal for exploration, while still taking advantage of UCB for the exploitation.

To use causal graph for exploration in R-UCB, we rely on four steps, as depicted in Figure 13.2:

1. Establish pairwise causal relation between observed context data by applying causal induction using the history.
2. Establish partial causal graph from ontological rules and other expert provided knowledge.
3. Assemble the causal graph from output of steps one and two.



(a) Probability tree statistically generated from the history. θ is the hypothesis that $L \rightarrow M$, oppositely $\neg\theta$ is the hypothesis that $L \leftarrow M$



(b) Probability tree after intervention. We intervened on variable M, supposed observed, thus true. It may now be unbalanced toward θ or $\neg\theta$

$$P(\theta|L,M) = \frac{P(M|\theta,L)\times P(L|\theta)\times P(\theta)}{P(M|\theta,L)\times P(L|\theta)\times P(\theta)+P(L|\neg\theta,M)\times P(M|\neg\theta)\times P(\neg\theta)}$$
$$P(\theta|L,M) = \frac{\frac{2}{5}\times\frac{5}{6}\times\frac{1}{2}}{\frac{2}{5}\times\frac{5}{6}\times\frac{1}{2}+1\times\frac{2}{6}\times\frac{1}{2}} = \frac{1}{2} = P(\theta)$$

(c) Before intervention, $P(\theta|L,M) = P(\theta)$, the probability tree is, by construction, balanced. Consequently and similarly: $P(\neg\theta|L,M) = P(\theta)$.

$$P(\theta|L,\hat{M}) = \frac{P(\hat{M}|\theta,L)\times P(L|\theta)\times P(\theta)}{P(\hat{M}|\theta,L)\times P(L|\theta)\times P(\theta)+P(L|\neg\theta,\hat{M})\times P(\hat{M}|\neg\theta)\times P(\neg\theta)}$$
$$P(\theta|L,\hat{M}) = \frac{1\times\frac{5}{6}\times\frac{1}{2}}{1\times\frac{5}{6}\times\frac{1}{2}+1\times1\times\frac{1}{2}} = \frac{5}{11}$$

(d) After intervention, $P(\theta|L,\hat{M}) \neq P(\theta)$, the probality tree is now unbalanced. $P(\theta|L,\hat{M}) < P(\theta)$, thus the tree is unbalanced toward $\neg\theta$, meaning: $M \rightarrow L$

Figure 13.3: Example of causal induction

Example of causal induction between two variables from H_{vocal_alert} 13.1. Let L be the context data (arthur isLocatedIn livingroom) and M be (arthur isDoing music). The causal induction aims to determine the causal relation between M and L.

4. Extract context data that are identified as failure cause thanks to the generated causal graph.

Let us review these steps more specifically.

13.5.1 Causal Induction

To select a context data that might be a cause of a failure, we aim to use a causal graph that represents the causal relations between context data, including the task failure. The first step toward it is to establish causal relations between pairs of data. The causality is defined as follows:

Definition 33. *Causality:*
A context data X causes another Y ($X \rightarrow Y$) if the object of Y is dependent, or partially dependent, of the object of X.

For example, the context data M = *(arthur isDoing music)* causes L = *(arthur isLocatedIn livingroom)*, as from Nono's experience, Arthur always listens to music in the living room. Note that, in this contribution, we do not deal with numerical value, such as the room sound level, this is left as a perspective. Nevertheless, numerical context data can be easily transformed and/or used through rules, for example the sound level can be classified into 'low' or 'high' by rules and then used in causal induction.

To determine causal relations, we apply Ortega's Bayesian causal induction [130] on observed context data and the history. Bayesian causal induction relies on the usage and intervention on particular probability trees. In fact, if the causal induction aims to determine the causality relation between two context data M and L, it will create and use a probability tree that models two hypotheses: $\theta : M \rightarrow L$ and $\neg\theta : L \rightarrow M$. An example of such tree can be found in Figure 13.3a, notice how the tree is constructed, with θ and $\neg\theta$ forming the upper nodes and M and L being inverted on the left and right branches, representing $L \rightarrow M$ and $M \rightarrow L$. By intervening on the tree and evaluating how the intervention unbalances the tree, we can determine the causal relation between the two context data. Be aware that the intuition of this approach can be difficult to grasp, do not hesitate to refer to this link[3] for further and less formal explanations.

The first and essential step is to construct this tree. It is actually built from statistical analysis of context data observed in situations stored in the history. In our case, the tree is constructed based on the history and the number of occurrence of each piece of context data according to the other. The two hypotheses θ and $\neg\theta$ are first represented with equal distribution: $P(\theta) = P(\neg\theta) = \frac{1}{2}$. Then, the tree is filled with the probabilities between the two questioned variables. Let us consider the history H_{vocal_alert} in Table 13.1 and the two context data M *= (arthur isDoing music)* causes L *= (arthur isLocatedIn livingroom)* whose causality relation is questioned. Let us say that X was observed two out of three times Y was observed in H, then $P(X|Y) = 2/3$ and $P(\neg X|Y) = 1/3$. In the history, it appears that M was observed two out of the five times L was observed, thus $P(M|L) = 2/5)$ and oppositely $P(\neg M|L) = 3/5)$. The same process is applied for any probability needed in the tree to represent the two hypothesis θ and $\neg\theta$. An example of resulting tree for our example is shown in Figure 13.3a. Note that $P(X|\theta) = P(X)$, for any variable X, when building the tree. Thus, for any X and any Y, $P(X|Y) = P(X|Y, \theta)$. This tree is, by construction, balanced. As expressed in Figure 13.3c, $P(\theta|L, M) = P(\theta)$, meaning we don't have clue toward one hypothesis or the other.

This tree does not learn us anything by itself. However, by analysing its "reaction" to the addition of causal information, we can infer causal relation. This is what we do by intervening on the tree. The intervention consists of setting a variable to a given value. In consequence, in the probability tree, probabilities of the value are now 1, and 0 for the non value. In our case, we intervene on a currently observed context data. For example, if M is observed in the live situation, we can intervene on it and is now referred to as $\hat{M}$, by intervening, we fix $\hat{M}$ to having been always observed, thus, for example $P(\hat{M}|\theta, L) = 1$. You can find the resulting probability tree in Figure 13.3b. By doing so, we introduced statistical asymmetry based on the observation. From there, we can try to induce the causal relation. To do so, we compute the probability of the hypothesis according to the two variables, in our case $P(\theta|L, \hat{M})$. Three cases are possible:

- $P(\theta|L, \hat{M}) > P(\theta)$: the tree is unbalanced toward θ, thus $L \rightarrow M$.
- $P(\theta|L, \hat{M}) < P(\theta)$: the tree is unbalanced toward $\neg\theta$, thus $M \rightarrow L$.
- $P(\theta|L, \hat{M}) = P(\theta)$: the tree is balanced, there are not causal relation between M and L.

We detail the calculus of our example in Figure 13.3d. As we can see, based on the situations in

[3] `http://www.adaptiveagents.org/bayesian_causal_induction`

H_{vocal_alert}, $P(\theta|L, \hat{M}) < P(\theta)$, thus we conclude that M causes L. Please note that, intervening on L would have given the same result, as $P(\theta|\hat{L}, M) > P(\theta)$ and $P(\theta|\hat{L}, M) = 1 - P(\theta|L, \hat{M})$.

After these steps, we now have the causal relation between the two context data M and L. The final process creates a two-nodes causal graph representing the computed relation. It is associated with a weight w that takes the value of $P(\theta|\hat{L}, M)$ if the tree was unbalanced toward θ and $1 - P(\theta|\hat{L}, M)$ if unbalanced toward $\neg\theta$. w is consequently always greater or equal to $P(\theta)$. This w does not represent the probability of the causality relation, but underlines how strong was the unbalance, thus giving an indication on the causal strength. Figure 13.4 show the resulting graph from our example.

This process is repeated for all pair of context data present in the live situation and carried in the situations stored in the history of a task. The failure, or success, of the task is considered as a context data, thus having relation with other context data. The intervention is performed according to observed context data in the live situation. In the end, a set of causal relation, represented as two-nodes graphs, is generated and will then be used to generate the causal graph. But before doing so, LEAF can take into consideration other sources.

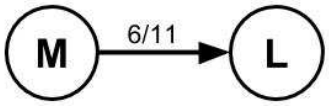


Figure 13.4: Output of the causal induction process between L and M

13.5.2 Static Knowledge

Causal induction is not the only way to determine causal relations between context data. Some knowledge is actually already available or can be specifically provided by an expert. Firstly, as we used ontologies for modelling the situations, we can use inference rules to infer new context data. These rules implicitly provide information about causal relation. In fact, a context data generated from others means it is dependent to, thus caused by, these others. Let us consider the rules: `(arthur isLocatedIn livingroom)` $\wedge$ `(nono isLocatedIn bedroom)` $\rightarrow$ `(arthur vocUnreachTo nono)`. This rule states that Arthur can be reached vocally by Nono if he is in the living room while Nono is in the bedroom, because of the distance between the two rooms. Thus, if L =*(arthur isLocatedIn livingroom)*, B = *(nono isLocatedIn bedroom)* and U = *(arthur vocUnreachTo nono)*, the rule implies the relation: $L \rightarrow U$ and $B \rightarrow L$. These relations are also associated with a weight w that is set to 1, as we consider this causal relation to be reliable. Thus, it conducts to the small causal graph depicted in Figure 13.5.

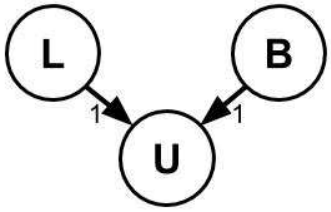


Figure 13.5: Example of causal sub-graph created from the semantic of an ontological rule

On top of that, an expert can provide predefined causal graph. Even if an expert can't foresee all causal relation, it can still provide some that can be useful and safe. Such graphs are stored in a dedicated knowledge base. Similarly to graphs inferred from rules, the weights are set to 1.

Alongside with the pairwise causal relations computed with the causal induction, the causal graphs carried through rules and expert knowledge are then gathered and assemble to a causal graph.

13.5.3 Assembling the Causal Graph

Once the causal relations between context data are set, the causal graph can be generated. As expressed earlier, this graph carries as nodes the context data and as edges the causal relation. The edges are weighted according to the result of causal induction or set to 1 if provided by static knowledge. The graph is simply generated by merging all the sub-graphs into one by matching the nodes in the sub causal graphs. Two nodes are merged if the context data they carry have the same subject, predicate, and object. In case of a contradiction of causal relation, the edge with the highest weight is kept, others are discarded. In the unlikely event of a contradiction with similar weights, all impacted edges are discarded. An example of a causal graph can be found in Figure 13.6. The outcome of the task, that is a failure, is a particular context data, in fact, the extraction process is based on the causal relation of other nodes toward this. Let us now see how the context data is selected in the causal graph.

13.5.4 Causal Graph Exploration

Once the causal graph is generated, we can analyse it to determine what could be the causes of the failure. As we aim to extract possible causes for a task failure, the causal graph is used by focusing on the context data related to the task outcome (F in Figure 13.6). The principle is to select nodes, in other words context data, that have a path to the task failure's node. In fact, there might be multiple possible causes, yet, as the user only validates a given number of causes, the aim of this step is to select the best candidates for validation. From a multi-armed bandit point of view, the aim is to select the candidates that are more likely to provide a high reward.

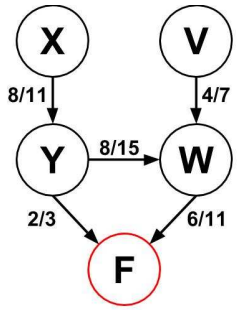


Figure 13.6: Example of a generated causal graph

To do so, the possible causes of failure are ordered according to the two following rules:

1. Priority is given to the vertex closer to the node 'task failure'.
2. Priority is given to the vertex with the highest path weight. The path weight is computed by multiplying the weights of all edges composing the path.

If multiple paths are possible for one vertex, the one with the highest path weight is used. For example, in Figure 13.6 the resulting order would be: Y, W, X, V.

With the vertices ordered, a last filtering process is executed. In fact, this cause extraction is part of the exploration, thus only the unknown causes should be extracted. Indeed, the ones that were already validated, or invalidated, are selected through the method previously described for exploitation. In consequence, all causes that are registered in the cause knowledge are excluded from the list. For example, let say context data Y is already in the cause knowledge, it is removed from the list that now consists of: W, X, V. Once the ordered list is filtered, the first context data is selected for extraction. This data will then be validated by the user alongside other possible causes extracted from exploitation or other instances of exploration. From there, LEAF continues the process previously

described, but has now a more accurate exploration that should allow the robot to learn quicker from its failures. This was verified through our experiments.

13.6 Experiments

In the experiments of LEAF, we aim to evaluate its ability to learn accurate failure causes from situations, and, consequently, its capacity to avoid tasks that are likely to fail. To do so, LEAF was confronted to several situations through simulations and was compared to a state-of-the-art method [157]. We aimed to measure how quickly and how efficiently is the learning of failure causes. LEAF was also implemented and tested with a robot. Let us review the experiments of LEAF.

13.6.1 Implementation

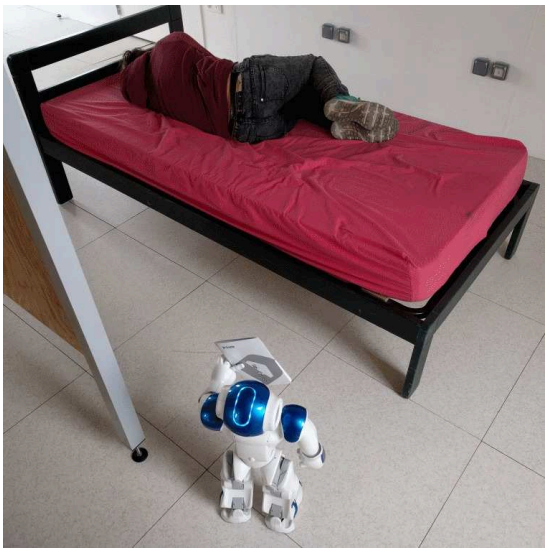


Figure 13.7: Example of scenario conducted with a Nao and LEAF

LEAF was implemented in Java and relies on Jena to manage ontologies. It was integrated and tested on a Nao Robot in the HadapTIC platform. Nao ability to speak was used to interact with user for cause validations. It was integrated with the DHTN planner that uses the cause knowledge in its process. Videos of our tests can be found online[4] and Figure 13.7 illustrates them. For evaluating LEAF, we relied on randomly generated dataset. By doing so, we can simulate a large amounts of situations to properly test the learning. The core code of LEAF is available on Github[5].

13.6.2 Protocol

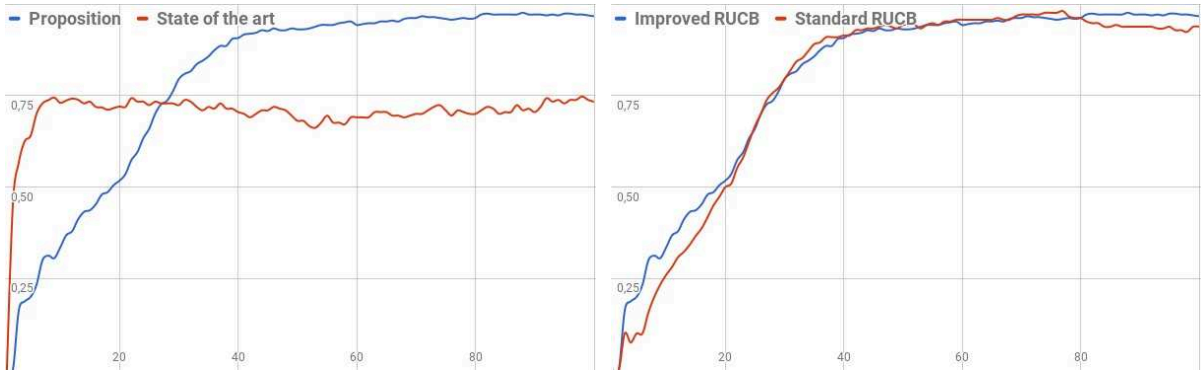
We evaluated our approach on a randomly generated dataset. The dataset consists of multiple situations that include various possible task failures. The generation engine actually create an ontology representing a situation by randomly generating context data as RDF triples from given patterns.

[4] http://nara.wp.tem-tsp.eu/what-is-my-work-about/leaf/

[5] https://github.com/Nath-R/LEAF

Patterns and generation probabilities are defined in a Java method. The idea is to see how our approach learns and prevents the failures. Each run consists of a generation of 100 situations. For each run, we measure the correctness of the failure prevention according to the number of encountered situations. A prevention is correct if the robot selected the most efficient task without failing. The process was repeated 20 times. We considered the user to provide 3 feedbacks at a time. Note the number of feedbacks influences the quickness of the learning of our approach, yet the tendency of the curve remains similar. The results are depicted in Figure 13.8. Each graph shows the evolution of the correctness of failure avoidance as the simulated robot gets more experience. Expectedly, the more experience the robot has, the less erroneous it is, yet the pattern of the learning curve varies according to the used method.

13.6.3 Results



(a) Comparison between our proposition and a state-of-the-art solution

(b) Comparison of our proposition with a standard and improved R-UCB

Figure 13.8: Correctness of task risk evaluation according to the number of situations encountered

Based on these measures, we compared our approach to a state-of-the-art solution similar to Sariel and Kapotoglu proposition[157, 86] in Figure 13.8a. We observe that, with the state-of-the-art method, the curve rises quickly before stabilizing. As the approach of Sariel and Kapotoglu is not limited by validations, it can identify many causes at once and reach quickly (after 10 situations) its maximum correctness of 75%. However, their approach makes some errors, making it identify wrong causes that are not corrected by user validation. This explains why it cannot go beyond 75% of correctness. As our approach relies on validation, the learning phase is longer as the robot has to ask the user multiple times. Indeed, after 10 situations, our approach achieves only 37% of correct task selection. However, thanks to the rich semantic representation and the user validation, our approach enables 95% when trained enough. This means that, once the robot has enough experience, it has only 5% chance to fail a task or select an inappropriate one, which is much lower than the 25% of the state-of-art solution. All in all, we can say our approach proves to identify and prevents accurately the failures, but at the expense of a longer learning time compared to the related work.

We also compared our approach with a standard R-UCB and with our improved version using causal graphs to assess its efficiency. Results are shown in Figure 13.8b. Our improvement enables a quicker correctness increase at first, reaching 30% of correctness after 8 situations, while the stan-

dard R-UCB reaches it after 12 situations. This proves the causal graph to be slightly more effective in finding possible causes during exploration. The more experienced the robot is, the more it relies on knowledge consolidation, hence, we expected the two versions to reach similar correctness as they should behave similarly. This is what we observed after 22 situations. Although it allows a better cause identification at first, the overall gain of the R-UCB improvement remains limited in LEAF. In fact, we observed that, although causal induction could pinpoint proper failure causes, it can also be biased and lead to erroneous cause extraction. Even if relying on high level context data, the causal induction remains a statistical approach and can be mistaken according to the experience. Indeed, with too few situations, causal induction can fail to determine causal relations. Nevertheless, even if limited, our improvement is pertinent as it allows the robot to be less faulty when it has little experience.

In conclusion, LEAF enables a very accurate learning of failure causes that allows to prevent almost all failures when trained. This is enabled by its ontological knowledge as well as its reliance on user feedbacks. However, this is also a drawback, in fact, as the number of user validations is limited, the learning is slower than in pure statistical approaches. In short, LEAF trades off time for accuracy. LEAF also improved R-UCB with causal induction. However, this approach is sensitive to bias, preventing LEAF to achieve ground breaking improvements. Yet, it enables a gain, in particular after a few situations, when causes are to be discovered through exploration.

13.7 Conclusion

In this chapter, we presented LEAF, a reinforcement learning approach to identify failure causes from experience. Similarly to the contributions presented in the previous chapters, LEAF relies on ontologies to model situations in the history. In other words, through LEAF, the robot has a highly semantic experience, allowing it to understand complex failure causes that are, for instance, inferred from rules or activity recognition processes. LEAF also includes the user feedbacks in its process to ensure the correctness of identified causes. To optimize its process, LEAF uses a multi-armed bandit approach, R-UCB, that allows it to select the most pertinent context data for user validation. However, R-UCB had to be adapted and improved, in particular for discovering new possible causes. In consequence, LEAF relies on causal graph generated from causal induction on the history. By doing so, context data can be identified more clearly as causes, but it also improves R-UCB and makes it more efficient in any context where there exists causality between the bandits. With this knowledge, a HTN based planner can be adapted to proactively avoid failures. In fact, for each possible decomposition, a confidence is computed and used to create plans that avoid tasks that are likely to fail.

By using ontologies, user validation and causal induction, LEAF enables a rich and reliable learning of causes failures, allowing to generate failure-proof plan. This what we aim to proof through our prototyping of LEAF. This prototype was tested on a real robot, but also through simulation. In fact, we simulated hundreds of situation and evaluated how LEAF is able to learn and prevent task failures. Results show that LEAF enables the robot an accurate learning of the failure, allowing it to prevent further failures. Furthermore, the causal improvement of R-UCB offers a small improvement when the robot is at the beginning of its learning. However, although LEAF is enabling a more

accurate learning than the state-of-art of reference, its learning time is longer, meaning that the accuracy comes at the price of a slower learning. This is explained by the fact that LEAF relies on reinforcement learning and user validation: in consequence, it needs more time to acquire experience, but also acquires a more accurate knowledge. A perspective is to study the possibility of combining two learning approaches to feature both quickness and quality of the learning. Nevertheless, LEAF enables the robot to enrich its planning knowledge from its experience.

Chapter 14

Conclusion

In this part, we presented two contributions of cognition and action that enables failure proof and context aware task planning for robots in smart environments.

First of all, we proposed a novel planner, DHTN. Our planning approach incrementally creates the plan while executing it, allowing to take into consideration the last changes of the environment. DHTN uses the formalism of HTN and extends it with notion of *status* to enable execution management on top of planning. Thanks to a hierarchy representation of tasks, DHTN decomposes a task into a sub-plan according to the last context observations. Through our experiments, DHTN as proven to encounter fewer failures and, consequently, to reach a goal with significantly fewer tasks than classical planning, thus preventing the robot to waste time and energy. On top of that, by dynamically computing and using only the relevant context data, it has a limited impact on the smart environment. Hence, by its dynamism and planning oriented context observation, DHTN is a task planner adapted to personal robots in smart environments.

However, this planner relies on a hierarchical planning knowledge provided by an expert. This knowledge can hardly be exhaustive, leading to failures as preconditions may actually be incomplete. Thus, we proposed LEAF, a reinforcement learning approach. Our learning approach acquires experience and uses it to identify task failures causes. Thanks to a multi-arm bandit modelling and the usage of ontologies, LEAF is able to accurately identify possibly complex failures causes. This knowledge of causes enhances the planning knowledge and allows to prevent failures in the future. Our experiments, relying on simulation, shows LEAF to enable a very accurate learning, but at the cost of a longer learning time. This can be improved Combining it with a pure statistical based approach could enable quick and accurate learning. This a way to explore. Another remaining issue is the uncertainty of context data in the task planning. In fact, uncertainty can lead preconditions to be wrongly evaluated, causing inappropriate plans. A possibility is to compute a confidence degree of the evaluation of preconditions and/or of sub-plans selection.

Yet, these contributions offer FAIRIE the ability to generate failure proof plan. All in all, FAIRIE enables robots within smart environments to acquire the context, to take a decision and to act. On the one hand, the decision is taken while dealing with heterogeneous context data and supporting five dimensions of uncertainty. On the other hand, the resulting actions have a limited risk of failures. The whole allows personal robots to be accurate and efficient, making them more suitable for taking care of users at home. But FAIRIE is to be confronted to reality. That is why all of our contributions

were prototyped and tested. But how close these tests were to real cases? And how FAIRIE can be set up, deployed and used in a real scenario? What improvements are required? This what we discuss in the last part of this document.

Part IV

FAIRIE Analysis

Chapter 15

FAIRIE and Prototyping

15.1 Introduction

In the thesis, we addressed and tackled problems encountered by personal robots in smart environment. As presented in the two previous parts, our contributions cover the challenges of perception, cognition and action. They can be used as part of a larger modular framework that is For An Integration of Robots in Intelligent Environments (FAIRIE). FAIRIE offers a set of tools for a robot to acquire data, take a decision and act in smart environments. Each of our contributions, that are FAIRIE modules, were implemented and tested through multiples scenarios. This was used to evaluate our works, as detailed in respective chapters, but also induced large engineering effort. In this chapter, we aim to discuss how these experiments were conducted. Through these experiments, we assumed the role of the designer[1]: from this experience, thus, we also aim to show how FAIRIE can be deployed for further experiments and/or for a real case application. First of all, it is important to review the tools and materials we used. Then, we describe how each module was implemented and how they can be assembled into FAIRIE. Lastly, we express how FAIRIE can be deployed and provide example from our experiments.

15.2 Experiment Environment

When conducting our experiments, we relied on multiple tools and materials. We used both a smart environment and a personal robot. We also relied on tools to create and manage our ontologies. Let us discuss those.

15.2.1 Smart Environment

To perform our experiments, we relied on the HadapTIC platform[2] as smart environment. This platform is located at Telecom SudParis, in the new Espaces et Technologies Ouverts pour l'Innovation des Laboratoires et des Entreprises (ETOILE)[3] building. In fact, this platform consists of a modular

[1] The "designer" refers to the person or team in charge of setting up and deploying robots and smart environments.
[2] http://hadaptic.telecom-sudparis.eu
[3] `https://www.telecom-sudparis.eu/recherche/etoile/`

(a) HadapTIC platform and its modular room

(b) Estimotes and one of the phone used

Figure 15.1: HadapTIC platform and devices used

room, as shown in Figure 15.1a, equipped with furniture, including beds and oven, and smart devices. It noticeably includes Legrand sensors, namely thermometers, motion sensors, door opening sensors, buttons and smoke detectors. On top of that, it carries multiple actuators that takes the form of controlled plugs or screens. Overall, there is between ten and fifteen devices deployed in the room at the same time. The platform also includes computers that can be used as servers if needed. The included devices mostly use ZigBee protocol with dedicated data format. Devices of HadapTIC were integrated within the m4iot platform[4]. The m4iot platform aims to provide a middleware for creating service within IoT. It relies on the notions of collectors, capsules, and applications. Collectors gather data from sensors, capsules manage and reason on those data, and applications are programs that uses and are trigger by the outcomes of capsules. m4iot models those elements in a distrusted architecture. It formats context data, defines communication formalism and uses CEP to generate high level context data or trigger applications. Furthermore, it adopts a publish-subscribe approach, allowing to query sensors with limited engineering effort. Hence, devices of HadapTIC were accessible through the middleware and eased our usage of smart environment.

Nevertheless, as the environment is open, we also added further devices in the environment. Such devices include Estimote[5] beacons, NetAtmo Welcome[6] and weather station[7], as well as mobile phones. Beacon were used to locate user phones. Phones sensors were also monitored, in particular their acceleration through their inertia unit. NetAtmo cameras aimed to locate people, however we encountered issues when deploying it. Finally, the weather station provides information about the air quality and temperature within the environment. To access these devices, dedicated API had to be used. Hence, we developed specific applications that used these API to access those devices. This require a noticeable effort of learning and development.

However, gathering data in physical environments can be difficult. Not only it requires a large engineering effort, it is also limited in the type of data available and the control we have on them; typically, it is hard to control the uncertainty of real devices. That is why we also used simulation

[4] https://www.telecom-sudparis.eu/wp-content/uploads/2017/04/M4IoT.pdf
[5] https://estimote.com
[6] https://www.netatmo.com/fr-FR/product/security/welcome
[7] https://www.netatmo.com/fr-FR/product/weather/weatherstation

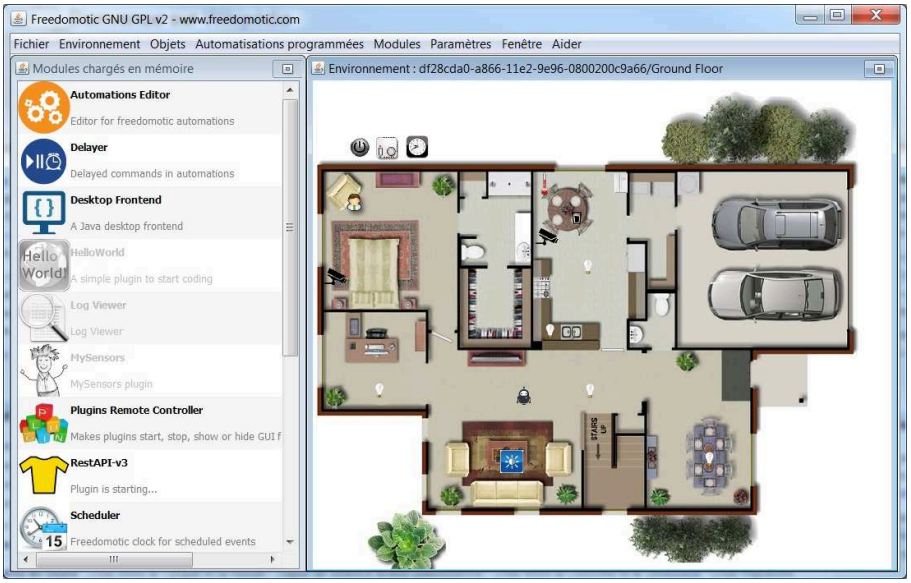


Figure 15.2: Freedomotic Interface

to evaluate our approaches. In order to simulate our environment, we used Freedomotic[8], an open source development framework used for managing smart spaces. Freedomotic allows to control virtual and physical devices. Thanks to its graphical interface, displayed in Figure 15.2, devices can be easily set, simple event rules defined, and plug-in used. In fact, it carries a basic event processing unit, a knowledge of the rooms, plug-ins and a REST API. One of the goals of Freedomotic is to simulate the environment before setting it up for real. Consequently, sensor values can be easily simulated. On top of that, it can be easily improved through plug-ins, developed in Java. The creation of plug-ins is made easy thanks to provided Maven archetypes, a simple API and manifests to define the plug-in. In our experiments, we used Freedomotic with custom plug-in and devices, including a virtual robot. It allowed us to generate context data and particular situations to test.

15.2.2 Personal Robot

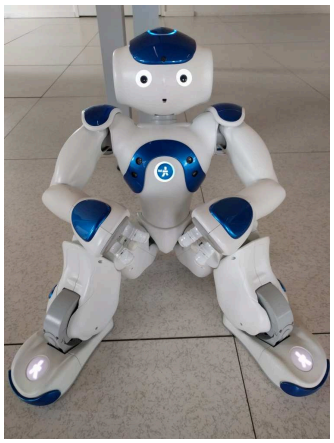


Figure 15.3: A Nao robot

[8] http://freedomotic.com/

In our experiments, we used the Softbank Robotics, former Aldebaran, Nao personal robots [63], displayed in Figure 15.3. Nao is a popular humanoid robot widely used by education and research communities. It is used in various fields including locomotion, Artificial Intelligence and Human Robot Interaction, in particular for Autism Spectrum Disorder. In fact, this robot offers many possibilities. It has 25 degrees of freedom, is equipped with two cameras, two sonar, an inertial unit, joint position sensors, tactile sensors, and more. On top of that, the robot is provided with a rich API, Naoqi, that makes the control trivial. The NaoQI API provides multiple tools and functionalities, such as speech recognition and speech generation. This C++ and Python API allows to make both embedded program, through cross-compilation, and remote scripting, running on a distant computer and controlling the robot through TCP. In our experiments, we designed multiple actions through Python scripts. An example of such a script can be found in Appendix C. A graphical development environment, Choregraphe[9], is also available, yet, we barely used it as part of our experiments. Nao can easily walk, even if the locomotion is relatively slow and energy consuming compared to wheel-based propulsion. A main limitation of Nao is its inability to perform SLAM, making robot localization complicated. We used alternatively two Nao H25 equipped with Naoqi version 1.14 and 2.1.

In order to control the robot, we used ROS. Robot Operating System (ROS)[10] is a famous middleware for controlling robots conceived by the company Willow Garage[11]. ROS is a powerful tool that enables hardware abstraction, reusability of functionality, communication between processes, package management and others. It particularly uses the concept of services, publisher/subscriber and topics. ROS mainly supports C++ and Python and is compatible with Linux, but a Java support has recently been released. We integrated DHTN with ROS, as well as the actions we defined for Nao as scripts. Actions were actually treated as services that the planner could query. By doing so, communication between our planner and the task to perform was transparent.

We also simulated the robot through Freedomotic. To do so, a plug-in was designed to interact with ROS. Thus, DHTN was managing the plan through ROS in a similar way for both simulated robot and Nao robot.

15.2.3 Ontologies

Our contributions largely relied on ontologies. In order to create and/or manipulate ontologies, we used Protégé[12] [55, 94]. Protégé is a famous open-source platform for ontology creation supporting various format, including OWL and RDFS. It can also be extended through plug-ins, such as FuzzyOWL2[13].

To manipulate ontologies within our implementations, we relied on Apache Jena[14], a free and open source Java framework. Jena allows to query ontologies, manipulate them, and perform reasoning through dedicated API. It also provides a formalism and an engine for inference rules.

[9] http://doc.aldebaran.com/2-1/software/choregraphe/index.html
[10] http://www.ros.org/
[11] http://www.willowgarage.com/
[12] https://protege.stanford.edu/
[13] http://www.umbertostraccia.it/cs/software/FuzzyOWL/index.html
[14] https://jena.apache.org/

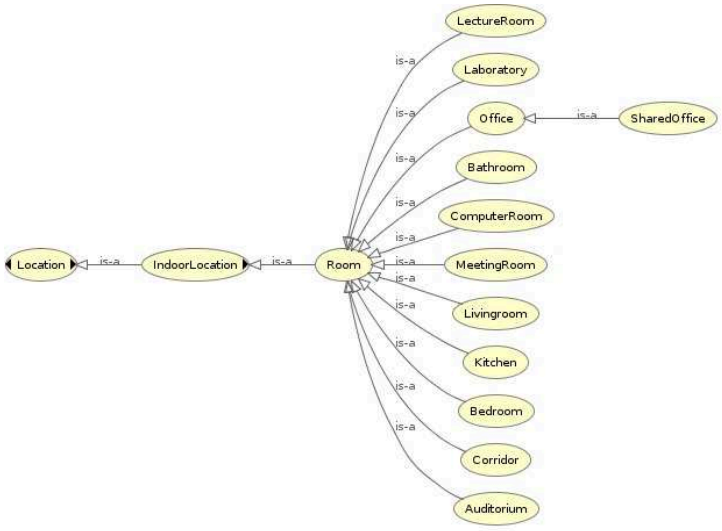


Figure 15.4: Excerpt of the structure of a background ontology

Thanks to Protégé and Jena, we designed and/or adjusted ontologies we used in our contributions. Multiple ontologies can be used, according to the environment and needs, however, as mentioned in Section 8.2.2, they must embed the following notions a minima:

- Class *Sensor*, and subclasses of *Sensor*.
- Data property *hasConfidence* associated to *Sensor* subclasses that expresses the reliability degree for a sensor.
- Data property *hadId* associated to *Sensor*, that links each instance of sensor with an identifier.
- Class *Activity*, that represents the user activity.
- A set of inference rules.

In our works, we mainly used an ontology improved from Rodriguez et al. contribution [151]. This ontology was initially used to model the context and detect human activity. Furthermore, it supports fuzzy logic by relying on fuzzy OWL 2 [25]. Consequently, it carries a rich variety of classes, data properties and object properties. In particular, it includes representation of locations, as seen in Figure 15.4, but also of devices and activities. This ontology was improved with classes and properties required in our contributions. Excerpts of the structure of this ontology can be found in Appendix D. By using this ontology, both context and background knowledge can be represented very precisely.

15.3 FAIRIE Framework

In this thesis, we conducted multiple contributions that composed a large framework that is FAIRIE. FAIRIE is composed by multiple modules, that are our proposals, but not only. Each module has a given role within the perception, cognition and action spectrum of problem. According to the environment, robot and needs, FAIRIE has to be configured and specified accordingly; that is how FAIRIE is a framework. In order to evaluate them, each contribution was implemented and prototyped. The

whole form a prototyped version of FAIRIE. Various tools were used, and a large effort was required to achieve this. Let us discuss those implementations. Then, we will see how they can be combined and assembled in various ways.

15.3.1 Perception: FSCEP

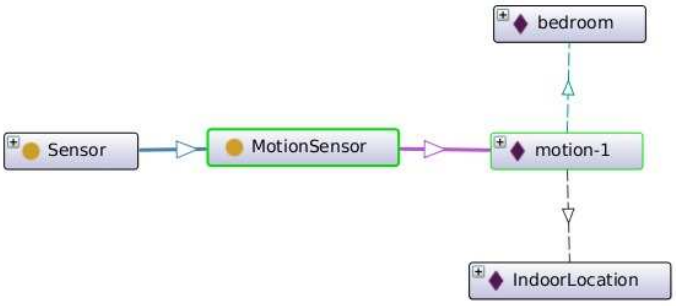


Figure 15.5: Example of motion sensor individual used in FSCEP experiments

In order to acquire the context under heterogeneous and uncertain sources, we proposed FSCEP, described in Chapter 8. FSCEP was implemented in Java and relied on Jena. It uses an agent-based approach and used Jade[15] consequently. Jade allows to define, create and manage agents. FSCEP also relies on Esper [50] from EsperTech: an event engine accessible as a software and/or library that enables management of events. It although provides a SQL-based formalism.

In this implementation of FSCEP, multiple agents are to be defined: one agent was related to one CEP rule. Each type of agent has its own Java code, multiple instances can then be launched through Jade. Note that all agents were deployed on a single machine. The role of agents is actually to separate the management of each rule. CEP rules are used to gather, filter and batch events, they are defined as strings embedded in the code of each agent. For example, these are two possible rules that are carried by different agents:

$$\texttt{Select events of } type\texttt{="motion" and } type\texttt{="identity" in time\_batch(1min)}$$
$$\texttt{Select events of } type\texttt{="temperature" in time\_batch(1min) where } value < 100$$

FSCEP uses a background ontology for semantization, any ontology that carries the minimal notions listed in Section 15.2.3 can be used. In our work, we used an ontology improved from the one proposed by Rodriguez et al. [151], presented in Section 15.2.3 and Appendix D. For our experiments we used Freedomotic and simulated a working activity scenario, consequently, we defined about ten individuals in the background ontology, including sensors, persons and rooms. An example of an individual is given in Figure 15.5. Obtained Results are described in Section 8.3.5. As a module, FSCEP uses, as input, pushed data, provided by interfaces, and as output provide fuzzy semantic complex event that carries fuzzy context data.

[15] http://jade.tilab.com/

15.3.2 Perception and Cognition: VARSeR

The vision of the robot can be used to collect a particular context data that is the user activity. As vision-based solution appeared to be inaccurate, we proposed a hybrid solution, VARSeR that combines vision-based recognition with ontological context knowledge and reasoning. The approach itself is described in Chapter 8. VARSeR relies on an already implemented vision-based solution [46]. The rest was implemented by Vincent Vassout during his four months training in the laboratory. The vision solution was deployed on the robot, while VARSeR itself was set up on a dedicated computer. Extensive experiments were conducted within the HadapTIC platform[16] and with a Nao robot. Multiple volunteers participated in these experiments. The detailed protocol and result can be found in Figure A. The resulting module uses as input a video stream and provide a set of weighted activities, that are context data.

15.3.3 Cognition: CAREDAS

In order to take a decision from the context knowledge, we proposed CAREDAS, a rule-based approach that identifies anomalous situations under uncertainty and determines a goal to pursue. CAREDAS is addressed in Chapter 9 and was implemented, including the MLN management. The prototype was designed in Java and represents more than 2100 lines of code. Again, it uses Jena to access a context ontology and any ontology that carries the minimal notions listed in Section 15.2.3 can be used. CAREDAS relies on rules that are carried in the MLN, as seen in Section 9.2: due to this particular formalism, in the current prototype, rules are directly inserted in the code. Using dedicated files to define rules is a solution, yet it was not relevant for our experiments. In order to evaluate this contribution, a dataset was acquired from the HadapTIC platform. It was acquired by performing 15 instances of a scenario, leading to an overall two hours acquisition and a dataset composed of 830 events, that is 55 events per scenario on average. Results can be found in Section 9.5. As input, CAREDAS accesses an ontological context knowledge that is filled from the output of perception modules. As a result, CAREDAS identifies anomalous situations and outputs a goal to reach if necessary.

15.3.4 Action: DHTN

DHTN is the task planner of FAIRIE. As seen in Chapter 12, our planner was designed to dynamically create and execute a task plan to take into account the last changes of the environment. On top of that, it uses only the required data for planning. DHTN was implemented in C++ and integrated in ROS. The core of the planner represents 3800 lines of code. It was tested through both simulation and with a robot within HadapTIC. When conducting those tests, it was essential to define the planning knowledge. In DHTN prototype, the knowledge was provided through custom files that defined methods and tasks. Two files are considered: one to define tasks, primitive and compound, the other to define methods. This second case is illustrated in Figure 15.6.

In our experiments, we considered multiple small scenarios, each consisting of ten to fifteen tasks

[16] The HadapTIC platform was not yet in the ETOILE building. IT consisted of two smaller rooms and five sensors.

```
main_meth
go_to_charge

go_to_door; open_door; leave_room; go_to_recharge_station; charge
--
meth_open_motor
open_door
DoorMotorised(Object:door)
ask_door_help
--
meth_open_help
open_door
inSameRoom(Entity:this, Entity:human)
ask_human_help
--
meth_open_manual
open_door

approach_door; push_door
```

Figure 15.6: Example of a custom file to define DHTN methods

Custom file used to define methods for a simple opening door scenario. It carries four methods for two compound tasks: *go_to_charge*, that is the goal task, and *open_door*. Methods have preconditions if required, and a list of sub-tasks.

for four to tens methods. Overall, to perform our tests, we had to implement 20 different actions for Nao. This was performed by creating scripts using NaoQi and integrated in ROS, as the one shown in Appendix C. Interactions between DHTN and action scripts are enabled by ROS. We also evaluated DHTN through simulation. To do so, a virtual robot plug-in was added to Freedomotic. This plug-in simulated the action of a robot and communicated with DHTN. Simulation allowed us to measure DHTN performance and led to the results described in Section 12.5. As part of FAIRIE, DHTN uses as input a goal, that can be provided by CAREDAS for example. From this, DHTN has the robot perform the required actions.

15.3.5 Action and Cognition: LEAF

As the planning knowledge is not exhaustive, failure may occur due to unexpected conditions. To solve that issue, we proposed a reinforcement learning approach based on experience and user feedback: LEAF, presented in Chapter 13. LEAF was implemented in Java and consists of 2500 lines of code. As it relies on a context ontology and stores the experience as ontologies, it uses again Jena to manage them. The rich understanding of the context is key to properly identify failure causes, thus, inference rules were used on top of the context ontology. Rules are provided in a specific file and are evaluated by Jena engine. An example of such rules can be seen in Figure 15.7. For each performed action, an ontology is created and stored as a Turtle[17] file. Note that we picked this format for readability, any other format supported by Jena can be used. Information about these ontologies, including the data and task they are related to are stored in a small local database. The cause

[17] https://www.w3.org/TR/turtle/

```
[isCloseTo:
    (?obj1 LEAF:isLocatedIn ?room),
    (?obj2 LEAF:isLocatedIn ?room),
    (?obj1 LEAF:hasPosition ?pos01),
    (?obj2 LEAF:hasPosition ?pos02),
    (?pos01 LEAF:x ?x1),
    (?pos01 LEAF:y ?y1),
    (?pos02 LEAF:x ?x2),
    (?pos02 LEAF:y ?y2),
    difference(?x1, ?x2, ?diffx),
    ge(?diffx, -50),
    le(?diffx, 50),
    difference(?y1, ?y2, ?diffy),
    ge(?diffy, -50),
    le(?diffy, 50),
    notEqual(?obj1, ?obj2)
    ->
    (?obj2 LEAF:isCloseTo ?obj1)
]
```

Figure 15.7: Inference rules used in LEAF experiments

Inference rules used in LEAF experiments to infer if two objects are close to each other, according to their relative position.

knowledge is also stored in that database.

LEAF works in close relation with DHTN, as it allows to enhance its knowledge. Hence, it interacts with DHTN through an interface with ROS. This allows DHTN to trigger LEAF processes and for LEAF to provide access to the cause knowledge. Yet, other planners can be used with LEAF, in the same manner as DHTN does. LEAF includes the user in its process to use his/her feedbacks. This was performed through a small Nao behaviour designed as a Python script using NaoQi. This script has the robot ask a question and waits for the user answer. The interaction between this behaviour and LEAF is again enabled by ROS. LEAF was tested both through simulation and a physical robot. Simulation was performed by automatically generating random situations. The generation process, and obtained results, are described in Section 13.6. Within FAIRIE, LEAF needs as input a context ontology. As output, it provides a cause knowledge that enhance the planning knowledge.

15.3.6 Assembling FAIRIE

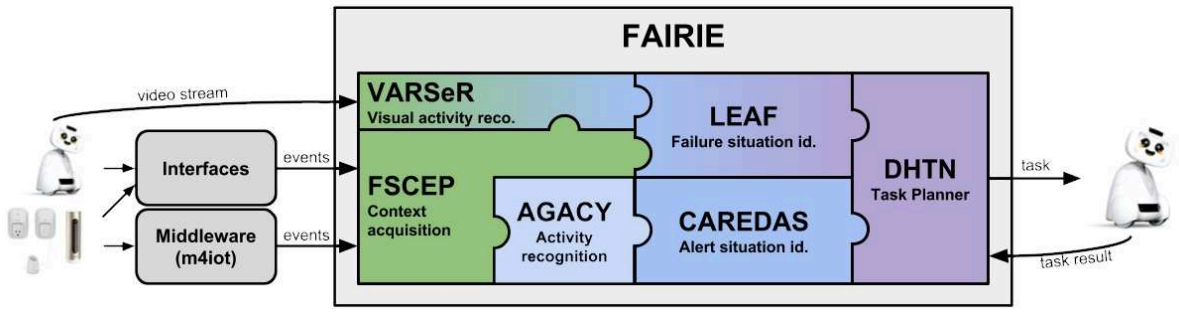


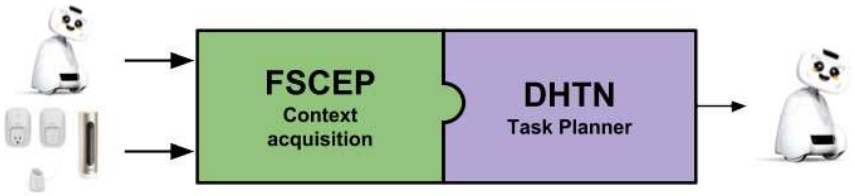
Figure 15.8: FAIRIE Architecture

FAIRIE is composed by modules that are our contributions. In fact, they can actually be chained together by matching inputs and outputs. As we have seen, all of them were implemented, hence, FAIRIE is implemented through them. Yet, every module can operate on its own, as long as proper inputs are provided. This enables FAIRIE to be configured in different ways to use, or not, modules according to the needs. Furthermore, contributions from other researchers can also be integrated within FAIRIE. For example, in our work, we considered the AGACY monitoring [161, 162] proposed by Hela Sfar. More generally, FAIRIE can use, and be used by, external tools.

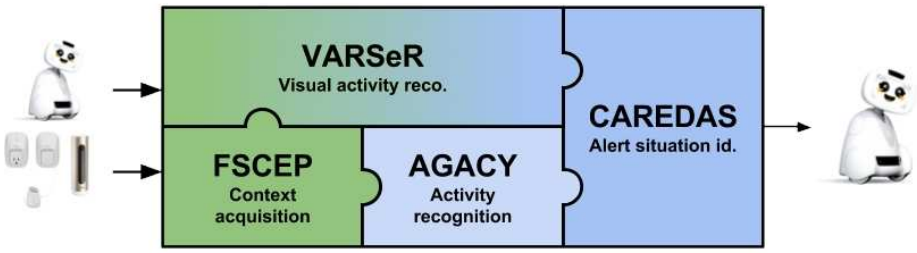
When combined, modules form a complete solution for robots in smart environment that cope with action, perception and action. The resulting architecture of FAIRIE, with all its modules, is displayed in Figure 15.8. A more detailed architecture of FAIRIE that includes interactions between modules and knowledge representations can be found in Appendix B. FAIRIE is a set of modules that interact according to their respective inputs and outputs. In our experiments, communications between modules was ensured through UDP communications and through data sharing, in particular for the context ontology. Yet, a middleware can be used to handle those exchanges, typically, LEAF and DHTN interact with each other through ROS. ROS usage in FAIRIE, mostly around DHTN, is displayed in Appendix B. By doing so, communications are transparent and normalized. Furthermore, other features of ROS, including interoperability and process control, make it a suitable tool to manage modules. This is why we aim to make FAIRIE fully compatible with ROS. As input, FAIRIE uses both video streams and events. While video streams are directly used by VARSeR, events need to be properly formatted to be used. Hence, sensor data are transformed through specific interfaces. For example, in our experiments, we designed mobile applications dedicated to acquiring and format data from specific sensors. Middleware can also be used: FAIRIE was indeed integrated with the m4iot middleware. To do so, we designed a FAIRIE capsule within m4iot. Hence, FAIRIE can access standardized context data provided by any m4iot collectors. More details about m4iot integration can be found in Appendix B. As output, FAIRIE, through DHTN, makes the robot perform specific tasks. As mentioned earlier, this is performed thanks to ROS. Yet, inputs and outputs of FAIRIE can vary according to environments and robots.

In fact, one of the most interesting feature of FAIRIE is its modularity: it can be configured according to the needs and possibilities, according to the robot and the smart environment. At minimum, FAIRIE can be configured with simply two modules, as depicted in Figure 15.9a. In such configuration, the context is acquired to feed the task planner, which is enough for basic behaviour. Similarly, another minimal configuration consists of FSCEP associated with CAREDAS for alerting exclusively. These basic functionalities can be enhanced with other modules to provide a complete context knowledge. For example, if the robot is equipped with cameras and has enough computer power, it can use VARSeR and AGACY for activity recognition. Thus offering more data for CAREDAS to use to detect anomaly situations, as shown in Figure 15.9b. From there, with this complete knowledge, the robot can provide a proper response by adding DHTN in the configuration of FAIRIE. Lastly, for the planning to avoid failure, LEAF can be used to learn from experience and enrich the knowledge of DHTN. The whole forming the configuration displayed in Figure 15.9c. On top of that, other external modules, matching the inputs and outputs of ours, can be integrated in FAIRIE configuration. Similarly, each module can be used as its own out of FAIRIE.

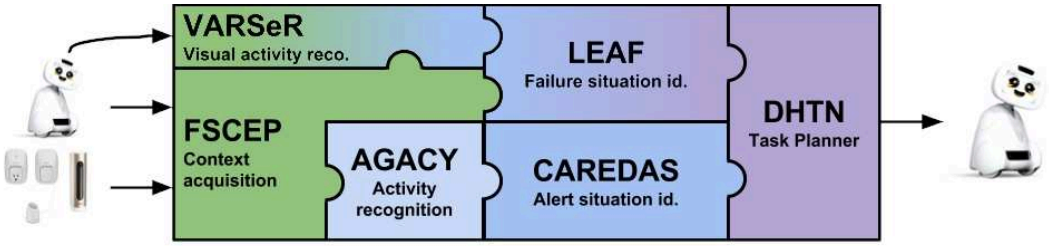
In brief, depending on the application, FAIRIE will be configured in a certain way, consequently, each module is set up in accordance to the environment. Deploying FAIRIE is actually not trivial and requires a designer to prepare modules properly, as we will discuss is the coming section.



(a) A minimal configuration of FAIRIE



(b) FAIRIE configuration for a complete context knowledge. In this configuration the robot only performs alerts



(c) FAIRIE configuration using all presented modules

Figure 15.9: Examples of possible configurations of FAIRIE

15.4 Deploying FAIRIE and Setting up Scenarios

Deploying FAIRIE is not an instantaneous task. In fact, many modules rely on specifications, thus on knowledge that needs to be provided by an expert. Most of this knowledge is dependent on both environments and robots, and consequently is to be provided when setting up FAIRIE in a home. Through our experiments, we experienced how the integration of modules can be performed. Hence, we defined rules, ontologies and others, that were required by our techniques. From this experience, we propose a procedure to follow in order to set-up new experiment scenarios or deploy FAIRIE for usage. We then illustrate this procedure on scenarios we followed during our experiments.

15.4.1 Procedure

In order to deploy FAIRIE, we propose a six-step process for a designer to follow to prepare FAIRIE.

1. **Selecting a Configuration:**
   As seen in the previous section, selecting a configuration for FAIRIE is the very first step of deployment. Hence, according to the characteristic of the environment and the needs of the user, the designer should select which modules to use. For example, we tried very simple and specific configurations for our experiments, such as a configuration only including FSCEP and CAREDAS. The designer can also select external modules that could be used, for example, he/she may add a social module for Human Robot Interaction. In that case, the designer should also ensure the integration of such modules in FAIRIE.

2. **Setting up Sensor Interfaces:**
   Acquiring data from sensors is one of the most important step. As we previously saw, sensors need to be interfaced with FAIRIE. Middleware, such as m4iot, can be used as interface. However, not all devices are using middleware or uniformed protocol. Hence, in most cases, implementing specific and dedicated interfaces is required. In our works, we implemented mobile applications to interact with beacons and weather stations. This step can be very demanding in term of engineering effort and add difficulties to the setting up of FAIRIE. In case of simulation scenarios, this phase consists of defining how data are generated and/or acquired. The designer should define what dataset to use or the characteristics of an automated data generation.

3. **Providing Ontologies:**
   Multiple modules rely on ontologies. There are two types of ontologies: context ontology and background ontology

   - First, the context ontology, that carries context data, needs to be set. The designer should provide the structure, hierarchy of concepts and properties used to model the context. In our contributions for instance, we used a base ontology [151] and improved it by adding key concepts as described in Section 8.2.2.
   - Then, the background ontologies should be provided. This ontology matches the structure of the context ontology, including the required concepts. However, it carries extra information about the environment used in some process, in particular in FSCEP. The designer must for instance include sensors with their identifier and their confidence value.

   Providing such ontologies and information requires the designer to have a strong knowledge of the environment FAIRIE is being deployed in.

4. **Writing Rules:**
   Several modules, on top of ontologies, rely on rules. Rules take different forms and purposes.

   - CEP rules are used to filter and batch events. Esper can be used to define such rules.

- Ontological inference rules are used to infer new context data, in particular in VARSeR and LEAF. Such rules can be defined using Jena formalism and engine.
- Rules are defined through MLN for modelling anomalous situations in CAREDAS.

For all of them, rules should be provided by the designer. Some modules, in particular CAREDAS or FSCEP, have rules at their core, consequently, rules should be carefully designed. Naturally, our experiments conducted us to design multiple rules. An example of an inference rule used in LEAF experiments can be found in Figure 15.7. However, we only considered a limited number of rules: in more real scenario, tens of rules should be provided which is, again, demanding in term of effort.

5. **Specifying Planning Knowledge:**
   If a planner, in our case DHTN, is selected in the configuration, the planning knowledge is also to be provided. Such knowledge carries actions, methods and preconditions. In our prototype of DHTN, this information was stored in a custom file. In more general cases, it is likely to be provided as PDDL files. Naturally, the planning knowledge directly depends on the robot and should be adapted consequently.

6. **Maintaining:**
   As the environment is open and may change over time, maintenance is essential to keep FAIRIE suited to it. This implies altering or adding new ones, adding background knowledge, or new tasks in the planning knowledge. Adding a new device also requires a manual integration through specific interfaces.

15.4.2 Deployment Scenario

As seen in previous chapters, our contributions were evaluated through multiple experiments based on various scenarios. On top of those, we also performed other tests to stress our implementations. In order to set up those experiments, we had to deploy and use multiple modules of FAIRIE. By doing so, we experienced the deployment of modules in various scenarios. Even if we considered specific modules, this is a good illustration on how FAIRIE can be deployed. Let us see how we prepared FAIRIE modules, according to the aforementioned procedure, for three scenarios.

Scenario 1: Detecting a Risk of Fire

Let us consider the scenario we expressed in CAREDAS. In this scenario, the user is cooking and then forgets it and perform another activity. FAIRIE should be able to detect the anomalous situation. This scenario is closely related to the one used for evaluating CAREDAS, but is performed directly rather than through a dataset.

Firstly, for this scenario, a simple FAIRIE configuration, composed of FSCEP and CAREDAS, is selected. FSCEP acquires the context that is then analysed by CAREDAS. Secondly, we prepared the sensors and their interface to FAIRIE. We used the HadapTIC platform and its sensors. Around ten sensors where considered, including motion sensors and buttons, that were used to model the

oven. They were accessed and formatted through a specific middleware[18]. We also used beacon through a dedicated mobile application. Thirdly, background and context ontologies were set up. We used the ontology presented earlier in Section 15.2.3. Then, rules are to be defined. First CEP rules were written. Two CEP rules were considered, one focused on location related events, while the other gathered the status of the oven. Consequently, two agents carrying those rules were defined in FSCEP. Anomalous situations were also defined in CAREDAS. In this scenario, the two rules used to detect the anomalous situations due to a risk of fire were created directly in Java. Planning and maintaining were not considered.

Through these steps, the two modules allowed to infer if the situation was anomalous or not. The robot can then intervene by warning the user.

Scenario 2: Going through a Door for the Robot

DHTN was tested in a simple opening door scenario. In that scenario, the robot decides to go to its recharge station as its battery are soon depleted. To do so, it has to go through a door and has two options: open it by itself, which is tough for it, or ask if a user is in the room. We used this test to see if DHTN was able to adapt to context changes, such as a user entering the room. Videos of these tests can be found online[19]. This scenario mainly focusses on the planning module and its specification.

In these experiments, only DHTN was considered. As for sensors, we relied again on the HadapTIC platform and its devices. Three sensors, including motion sensors and an opening detector, were used and accessed through a dedicated middleware. Acquired data were directly provided to DHTN through ROS. Actions for Nao were also implemented as scripts. We used as many scripts as tasks, that is to say 10 scripts. Neither ontologies nor rules were required. However, planning knowledge had to be provided. To do so, we provided two files defining tasks and methods, as shown in Figure 15.6. We defined 10 tasks, both primitive and compound, and 4 methods. From this preparation, the robot could use DHTN and perform actions to go through the door. The robot adapted to the entrance or departure of the user while it had already started the plan.

Scenario 3: Learning that User Inability to Hear the Robot

To test LEAF ability to learn from failures and to take into account user feedback, we designed a scenario where the robot is asked to remind the user to take its medicine but fails to do so. We considered the cause of the failure to be the user listening to music, thus not being able to hear Nao talking to him/her. We tested this scenario in the HadaptTIC platform with a Nao robot. Videos of this scenario can be found online[20].

Firstly, two FAIRIE modules were used together, DHTN and LEAF. Once again, we used HadapTIC sensors through a dedicated middleware. We used mobile phones, computers and motions sensors in those tests. Acquired context data were formatted and stored in a context ontology that was then used by both LEAF and DHTN. Thirdly, this ontology was defined. We used a simplified ontology

[18] Prior to m4iot.

[19] `https://drive.google.com/file/d/0B-G4jmh6-FuTVXVDYW92MGNXSjA/view?usp=sharing`

[20] `https://drive.google.com/open?id=0B-G4jmh6-FuTYUw3Wm1wZEZIMGs`

that carries the minimal concepts defined in Section 15.2.3. The created ontology had 25 classes and 21 individuals. Then, we defined inference rules that were used to infer new context data. We designed two rules, such as the one in Figure 15.7, provided in a specific file. Lastly, we defined a simple planning knowledge. We defined 6 tasks and 3 methods in custom files. Each operator, related to primitive tasks, is associated to a Nao action script. The planning knowledge includes a specific task that aims to collect user feedback as part of LEAF process.

15.5 Conclusion

In order to evaluate and stress our contributions, we conducted multiple experiments through both simulation and physical tests. Those experiments not only allow to validate each of our proposals, but also to tests them in various scenarios. In this chapter, we provided more details about our implementations and the experiment environments. Our contributions are composing the FAIRIE framework, thus, we then detailed how modules can be assembled in different ways according to the needs. As each module was prototyped and validated, FAIRIE was consequently prototyped as well. However, as it uses a lot of specified knowledge, including various rules and ontologies, deploying FAIRIE is not trivial. From our experience, we defined a six-step process to set-up FAIRIE. We then illustrated it through examples of deployment in three scenarios we used in our experiments. Even if those implementations were enough to validate our approaches, some consolidations can be performed. In fact, some features are not conveniently designed. For example, CAREDAS rules should be defined without altering the code and modules can be integrated in ROS, making their management and interaction easier. With such consolidation, FAIRIE can be more extensively tested in possibly more complex scenario. This is one of the main short-term perspective of this. Indeed, each module was validated and FAIRIE has proven its capability to face challenges of perception, cognition and action encountered by personal robot in smart environment. This consequently marks the end of this thesis.

Chapter 16

Thesis Conclusion

16.1 Introduction

As personal robots and smart environments are emerging, this thesis aimed to explore the computer science aspects of using both for domestic services. Challenges were numerous, ranging across perception, cognition and action. By proposing multiple contributions on the whole spectrum of problems, we acquired a global experience of data management for personal robots and/or ambient intelligence. From this understanding, we pointed out two main conclusions for research around robots in smart environments: the need for both learning and reasoning, and the diversity of impacting domains. These conclusions are worth considering for researchers and future works about personal robots in smart environments. In fact, our works open up multiple perspectives, including both improvement of our proposals, and new research axes. Let us discuss those in this final chapter.

16.2 Thesis Contributions

For robots in smart environments, perceiving, thinking and acting can be particularly complicated, consequently the research community is active around those problems. However, through our analysis of the state-of-the-art we pointed numerous open research challenges. Perception and cognition are subjected to heterogeneity of data, evolution and uncertainty. Yet, existing learning-based, specification-based or hybrid techniques, only solve a part of these challenges. Uncertainty is particularly topical challenge as no technique is able to cope with the five dimensions that are imprecision, inaccuracy, freshness, contradiction and incompleteness. When acting in a home environment, robots can encounter task failures, thus failing their plans. In the literature, most approaches react to failures by regenerating the plan, however, this is not a satisfactory solution as the robot does not avoid the failure itself. Yet, only a few works tried to proactively avoid failures.

Overcoming those limits is essential to enable efficient robots in smart environments. In this thesis, we proposed the following contributions to tackle them:

- **Perception - From Sensors to Knowledge** For context acquisition over event based sources, we proposed a new tool combining CEP, ontologies and fuzzy logic. FSCEP is able to cope and/or model four uncertainty dimensions, namely accuracy, contradiction, precision and

freshness. FSCEP allows to provide fuzzy context data that can then be used in accordance in cognition. However, as not all sensors are compatible with an event-based approach, thus, we also explored using robot cameras for activity recognition. However, this approach provides inaccurate and imprecise results. We proposed a refinement of the visual activity recognition process to provide a more accurate recognition. By using an ontological context knowledge, VARSeR is able to correct the visual recognition. It provides a set of possible activities associated with probabilities.

- **Cognition - From Knowledge to Decision:** Identifying anomalous situations is essential for the robot to take a decision to intervene. To achieve such a feature, we used MLN alongside with an ontological knowledge. By relying on MLN, that combines formal logic and Markovian modelling, CAREDAS can cope with completeness and can take into account other uncertainty dimensions modelled through weights. Hence, combined with fuzzy context data provided by FSCEP, five dimensions of uncertainty are supported. CAREDAS is able to detect situations were the robot should intervene and help it take a decision.

- **Action - From Decision to Actions:** In order to reach goals from its decision or orders, the robot can use a task planner. Yet, the generated plan can become outdated as the context varies, causing the robot to fail and perform wasted tasks. To avoid that issue, we proposed a novel planning approach based on HTN that combines acting and planning, thus taking into account the last changes of the environment. Consequently, DHTN monitors the execution and is able to efficiently use external devices, for sensing and acting.

- **Cognition/Action - From Actions to Experience:** Not all failure causes can be foreseen in the planner knowledge, hence the robot is likely to encounter failure. We proposed a learning approach to identify failure cause from experience and enrich the planning knowledge to proactively prevent further failures. Our approach includes the user for validation and is modelled as a multi-armed bandit problem. We used and improved R-UCB with causal graph generated by causal induction. LEAF relies on a rich ontological context knowledge filled by using our other tools. This leads LEAF to be able to accurately learn and prevent failures.

Each contribution was implemented and tested on its own. Even if they solve their particular challenges and are independent, the contributions presented in this thesis are part of a larger framework: FAIRIE. FAIRIE is a modular framework that handle perception, cognition and action for robots and ambient intelligence. It can be configured differently according to the possibilities of the robots and environments. Furthermore, it can use other external propositions, such as AGACY. FAIRIE can be seen as the final resultant of this thesis. While going toward this global framework, we acquired a profitable experience. Experience that can be used for future works.

16.3 The Need for Learning and the Need for Reasoning

In our contributions, we used multiple times learning techniques as well as reasoning techniques, mostly based on specified rules. Each approach has its strengths for robots in smart environments.

On the one hand, specification techniques, including reasoning methods, enable controlled intelligence for the robot. In fact, thanks to the specified knowledge, the robot can immediately understand, think and act, without a long and possibly erroneous learning phase. On top of that, they are usually compatible with complex and heterogeneous data modelling, enabling a complete context understanding. For these reasons, we intensively used specification techniques in all levels, for example in FSCEP, CAREDAS or DHTN. Yet, as environments are open and various, it is impossible to specify everything, for every case. On the other hand, by using learning techniques, robots, alongside ambient intelligence, can adapt their tools to the environment. With such an approach, the robot can understand constraints of any environment and adapt to the evolutions, which is an essential feature for domestic applications. VARSeR and LEAF do take advantage of this strength. Yet, learning techniques lack depth of understanding and would require a strong learning phase if used by themselves.

In the end, both learning and specification based approaches are required for a personal robot within an ambient intelligence. The specification knowledge provides a base while the learning allows to adapt to specific constraints of the environment. In this thesis, we used both and proposed multiple combination approaches:

- VARSeR combines a learning and vision based activity recognition system with ontologies and rules.
- CAREDAS uses MLN, a combination between formal logic and markov models, and ontologies, again with rules.
- LEAF is used to learn failure causes and enhanced the specified knowledge of DHTN.

However, even if both learning and specified reasoning are essential, combining them is not necessary relevant. In fact, as pointed out in our experiments of VARSeR, even if enhancing the learning visual process with reasoning improved the activity recognition process, the reasoning approach seems to be enough by itself. Indeed, in some cases, reasoning if more efficient than learning, and oppositely. Typically, in the case of VARSeR, when devices related to an activity are available and provide data, the reasoning performs better than the learning approach. However, in some cases where the user is out of range of devices, the robot can detect the activity of the user based on the gestures it has previously learnt. In brief, it appears more relevant to use learning and reasoning techniques in complement to each other rather than performing the same role. This is, for example, what we proposed through LEAF and DHTN, where LEAF specifically focuses on learning from experience and then interacts with DHTN to enrich its knowledge.

Hence, we point out that both learning and specification techniques should be used, but not for the same purpose, as one is likely to be neglected, thus wasted. We state **both approaches should be used complementary**, and we believe that **combination of specification-based and learning-based approaches for a same objective is not promising**. In fact, an efficient way for a robot in a smart environment is to first rely on reasoning approaches and specification, provided by expert, and then to enhance its knowledge through learning. In other words, the robots should have a base intelligence, through specification techniques, and then rely on learning techniques to refine its knowledge to the context.

16.4 Robots in Smart Environments: a Multidisciplinary Field

The challenges addressed in this thesis are broad and various. Robotics and pervasive computing, including ambient intelligence, are by themselves, large domains with numerous sub-domains. In fact, proposing a robotic solution implies tackling a large spectrum of aspects and technologies, as we can see through projects such as Robot-Era [22] or CompanionAble [67], KnowRob [171] is also a good illustration as it use multiple techniques for representing the knowledge of the robot. By tackling both of them and focusing on the computer science oriented management of data, this thesis touched even more domains. To illustrate our point, in our works, we used various techniques from diverse domains. For example, we used R-UCB, that was initially designed for recommender systems; Complex Event Processing, widely used in business processing; and, ontologies, from semantic web domain.

In fact, research around personal robots in smart spaces touches diverse aspects, for perceiving, thinking, acting, and more: proposing a complete solution requires tackling numerous and various challenges, as we did in this thesis. In fact, each problem encountered by personal robots in smart environments require particular techniques and particular constraints. For example, acquiring the context is a much different matter than acting; yet, context acquisition and acting are not specific to our case and were used in various applications and fields of research. Hence, when contributing in the domain of robots integrated with ambient intelligence, it is important to consider works from various communities, not necessary directly related to robots or smart environments. Indeed, even if the context and applications are different in other fields, tackled problems may be equivalent to ours and proposed solutions may be applicable in our case. By having a global look over multiple fields, more doors open toward possible solutions.

On the other hand, it implies the literature to be more broadly reviewed to ensure the novelty of the approach: even if the approach is novel in a robotic context, it may have been applied to solve an equivalent problem in another field. In other words, when contributing toward robots in smart environment, it is important not to forget it is not the only research field. Moreover, an issue of a multidisciplinary field is the variation of terms and vocabulary. From a community to another, terms do not have the same definitions. Uncertainty is a good example, in fact, the dimensions and definitions are different in pervasive computing research and for data stream research; furthermore, it is closely related to data quality. Thus, it is essential to review state-of-the-art works with an abstracted look: the term maybe be different while the problems are the same. Lastly, positioning the contributions proposed toward personal robots in smart environments can be an issue. Indeed, it can be overlapping multiple communities, that is the case of LEAF for example. We believe it is important not to remain in one community, but to distribute the contributions toward multiple ones. By doing so, not only we can obtain diverse feedbacks from various points of view, but it also allows to share approaches that are not commonly used by a given community, thus reinforcing the strength of the contribution for the global research. Nevertheless, the novelty of the contributions should be ensured in all overlapped fields: simply reapplying a method in another domain is not a multidisciplinary contribution.

From these observations, we underline the importance of **defining precise objectives and chal-**

lenges when performing research around robots in smart environments. This allows to concentrate the reviews of the literature on limited aspects, while on various fields. With strongly divided objectives, the overall solution will be clear and properly structured. Then, more generally, an unlike some communities, **it is essential to be open to directions from all communities**, but it is also important to have a more abstracted view over it.

16.5 Perspectives

Taking feedbacks from experience into account is important for upcoming works. In fact, the works conducted through this thesis open up multiple doors for further research. We present possible perspectives we identified. Note that they are ordered by the distance, in both time and effort, from the current state of our works.

- **Framework and Experiments Consolidation:**
  As detailed in Chapter 15 and in previous parts, each of our contributions was implemented and experimented on its own. However, prototypes need some adjustments and consolidation in order to enable a perfect integration in FAIRIE framework. A first perspective is to consolidate our implementations and ensure their interactions between each other. For example, integrating all our modules in ROS would make interoperability and communications efficient and reliable. From this, we aim to perform further experiments, in particular with FAIRIE full deployed. To do so, we can rely on the HadapTIC platform, but also on the EVIDENT[1] platform, that is a complete apartment equipped with diverse smart devices including smart carpets, lights and others. This can be performed in a relatively short term.

- **Experiments in Real Conditions:**
  The previous perspective leads to a more larger, yet important, future work that is experimenting the whole FAIRIE framework in real conditions. The idea would be to deploy a personal robot and smart environments with a real person that would live with the system for several days, or even months. Doing such experiments is important to evaluate how the robot performs and how it is accepted by using our contributions. However, they are tough to set up and conduct. In fact, several phases are to be followed. Firstly, it would require to set up a proper protocol, defining the scenarios, duration and measurements. Secondly, robots and environment need to be set up and ready to interact with persons. Following, we would have to find volunteers to live for some days within our FAIRIE equipped environment. Finally, the last phase is to evaluate the performance and acceptance of the solutions by defining a survey to be provided to test subjects. The whole process is a matter of months and probably more than a year.

- **Contributions Follow-up:**
  As mentioned in previous parts, our proposals open research problems and new contributions can be conducted. Let us review them.

[1] `https://evident.telecom-sudparis.eu/`

- Learning Anomalous Situations: CAREDAS relies on specified rules to identify anomalous situations. However, as any provided knowledge, it may not be exhaustive, making the robot unable to react in possibly critical, yet unknown, situations. How to identify undefined anomalous situations? Similarly to LEAF, learning anomalous causality is a possible axis of solution. Causal induction can be used to infer causal relation between context data and anomalous situations. This would enable to learn situations causes and predict upcoming situations by observing presaging context data, directly or indirectly.
- Learning Quickly from Failures: As we saw, LEAF is able to learn and prevent task failures accurately, but at the cost of a longer learning phase compared to state-of-the-art approaches. Hence, how can the robot can learn quickly and accurately? Using both statistical based and multi-armed bandit based learning is an option. It would ensure quickness and quality of the learning at the same time. Combining LEAF with another statistical approach may not be relevant, however, using both alternatively seems to a good option. The idea is to rely on a statistical learning at first, to learn and avoid failures quickly, then to switch to a multi-armed bandit based including the user in the loop. In other words, the exploration phase is left to statistical learning. After, the multi-armed bandit approach allows to balance exploration and exploitation and achieve an accurate learning. The challenge is to determine when to switch from one to another. This can be performed by projecting the robot success rate.
- Planning under Uncertainty and Other Constraints: The context knowledge is important when planning, yet, as we extensively discussed, it can be uncertain. Furthermore, plan branches require various effort and times to perform. How can the robot can generate a proper plan with uncertain context data? How can the plan select the most efficient branch? An idea would be to propose a model cost for possible decomposition. In fact, such a cost model would allow to compute a cost based on, for example, tasks reliability, energy consumption and duration for tasks, as well confidence in decomposition. The idea is to not only select sub-plan based on preconditions, but also based on other metrics modelled in the cost. The confidence part of this model can be computed based on truth values carried in fuzzy context data, allowing to handle uncertainty.

- **Distributing FAIRIE:**
  All of our contributions have been prototyped, consolidated and evaluated. The following step is to make FAIRIE available for everyone. This would require a large engineering effort to rise the quality of our contributions implementations: they should not be prototype, but robust products that can be integrated with other major tools, such as Robot Operating System or KnowRob [171]. Furthermore, to make FAIRIE an all-in solution, multiple aspects, out of the scope of this thesis, are to be tackled. Firstly, as the robot and smart environment interact with the user, thus the social aspect is not to be ignored. Exploring the social aspect of the robot and use a data management approach to help the robot to be empathic and ethical is a way to explore. Then, the robot may be asked to collaborate with other robots or even users. Even if our tools are suitable for using smart devices, they are not adapted for multiple agents. For

example, our planner only consider one robot. A perspective is to improve tools, or propose new one, to enable multi-agents solutions. Numerous works explored collaborative robots or multi-agents solutions, such as HATP or Robot-Era, and can be use as a base or combined with our contributions. Lastly, in order to ease FAIRIE deployment, self-configuration is to be explored. This implies the robot to learn or adapt rules, to download required module on the fly, and to understand how to use and interact with smart devices. However, this is a major open research problem.

- **Toward Ubiquitous Robots:**
  Lastly, as the IoT is progressing, it is an important perspective to consider robots, not only in smart home, but as part as a larger environment with devices connected to the internet. Thus exploring the integration of robot in the Internet of Things and moving toward ubiquity is a promising, yet long way. Consequently, the usage of shared knowledge, downloadable modules and distributed intelligence, which can rely on Big Data technologies, are to be considered. In brief, a long term perspective is move toward robots that are not quartered in homes and that can provide ubiquitous services.

16.6 Personal Experience

Even if I have a computer science background, I had the occasion to discover robotics through projects and participation to international contests. This gave me a first contact with robotics and showed me how interesting and topical are robotic challenges. It also introduced me through the world of research, and I glimpsed what is it about, and what the possibilities. These experiences attracted and pushed me to pursue a Ph.D. Through this thesis, I wanted to explore robotics and use my computer science background to make robots closer to our homes. Today, it reaches it ends. It was a long uncertain way to make things move on: it was an adventure. An adventure rich in experience. It was a learning experience; through my works, I got to learn and see how robotics is broad and passionate issues are, from ethics to navigation. And while I focused on the on the intelligence of robots, I got in touch with numerous fields of research, including context awareness, task planning and others. It was a human experience; I got to meet and share with persons from all countries, domains and convictions. This enriched not only my work, be also myself as a human being. It was a fight experience; indeed, the thesis also brought tough moments when I, we, had to stand and keep going forward. Doubt and questioning were present all along, but never interrupted my path. It was a research experience; not only I did contribute toward more intelligent and autonomous robots, my research also taught me to have a new look on the world, on the problems that surround us. After these years, that is probably a new me that emerged. A new me that is still willing to make personal autonomous robots a reality. However, academic research hasn't seemed to be the path to continue on. Nevertheless, my tying with robotics has strengthened, and I hope to keep working toward the advent of personal robots in our (smart) homes.

Acknowledgements

The thesis was a long and stressful adventure, hopefully, many people helped me all along, and I am grateful to them.

Naturally, my first thanks come back to my supervisors, Amel and Béatrice, who had the patience and the will to push me forward over those almost four years. It is obvious that the thesis would not be what it is without their work and help, for the research, but also personally.

I would like to thanks also the jury of my defence, Dimitris Kotzinos, Claudia Roncancio, Patrick Reignier, Adriana Tapus et Sao Mai Nguyen, for their remarks, questions and the evaluation of my work. I am also grateful for the lab with all its members, that welcomed me.

I thank the team and the persons that worked with me: Alda, Badran, Fethi, Mossaab, Duong and others. I warmly thank my PhD-mates, and now friends, Monika (and Patryk), Katleen, Mai, Hela, Nabila, Tuanir, Wafa, Hanane, Kenza and Isma, that cheered me up, helped me a lot, and with whom I shared numerous memories.

Of course, I thank my parents, and my bro, for their support and care, that where more than necessary during this thesis. An important actor for me was my volleyball club that kept me alive, both physically and mentally: I am grateful to it and all its members. I also thank my friends, Johan, Rémi, Clément, Florent, JB, Antonin, Solène and Eleuda for all the good times that boosted my morale.

Finally, I would like to mention and thank the persons that orient me toward PhD. I first warmly thank Vincent Hugel, who gave me the passion of robots and provided me amazing experiences, with robots, projects, research and even travelling. I also thank Thierry Garcia who was essential in my decision of making a PhD.

Bibliography

[1] Abowd, G., Dey, A., Brown, P., Davies, N., Smith, M., and Steggles, P. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing* (1999), Springer, pp. 304–307.

[2] Ahmad, R., and Plapper, P. Safe and automated assembly process using vision assisted robot manipulator. *Procedia CIRP 41* (2016), 771–776.

[3] Alami, R., Chatila, R., Fleury, S., Ghallab, M., and Ingrand, F. An architecture for autonomy. *The International Journal of Robotics Research 17*, 4 (1998), 315–337.

[4] Alletto, S., Cucchiara, R., Del Fiore, G., Mainetti, L., Mighali, V., Patrono, L., and Serra, G. An indoor location-aware system for an iot-based smart museum. *IEEE Internet of Things Journal 3*, 2 (2016), 244–253.

[5] Anderson, D. T., Ros, M., Keller, J. M., Cuéllar, M. P., Popescu, M., Delgado, M., and Vila, A. Similarity measure for anomaly detection and comparing human behaviors. *International journal of intelligent systems 27*, 8 (2012), 733–756.

[6] Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living* (2012), Springer, pp. 216–223.

[7] Anicic, D., Rudolph, S., Fodor, P., and Stojanovic, N. Stream reasoning and complex event processing in etalis. *Semantic Web 3*, 4 (2012), 397–407.

[8] Arcangeli, J.-P., Bouzeghoub, A., Camps, V., Canut, M.-F., Chabridon, S., Conan, D., Desprats, T., Laborde, R., Lavinal, E., Leriche, S., et al. Income–multi-scale context management for the internet of things. In *International joint conference on ambient intelligence* (2012), Springer, pp. 338–347.

[9] Artikis, A., Etzion, O., Feldman, Z., and Fournier, F. Event processing under uncertainty. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems* (2012), ACM, pp. 32–43.

[10] Attard, J., Scerri, S., Rivera, I., and Handschuh, S. Ontology-based situation recognition for context-aware systems. In *Proceedings of the 9th International Conference on Semantic Systems* (2013), ACM, pp. 113–120.

[11] Augusto, J. C., Liu, J., McCullagh, P., Wang, H., and Yang, J.-B. Management of uncertainty and spatio-temporal aspects for monitoring and diagnosis in a smart home. *International Journal of Computational Intelligence Systems 1*, 4 (2008), 361–378.

[12] Augusto, J. C., Nakashima, H., and Aghajan, H. Ambient intelligence and smart environments: A state of the art. In *Handbook of ambient intelligence and smart environments*. Springer, 2010, pp. 3–31.

[13] Augustsson, S., Olsson, J., Christiernin, L. G., and Bolmsjö, G. How to transfer information between collaborating human operators and industrial robots in an assembly. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational* (2014), ACM, pp. 286–294.

[14] Azizzadenesheli, K., Lazaric, A., and Anandkumar, A. Experimental results: Reinforcement learning of pomdps using spectral methods. *arXiv preprint arXiv:1705.02553* (2017).

[15] Badii, A., Etxeberria, I., Huijnen, C., Maseda, M., Dittenberger, S., Hochgatterer, A., Thiemert, D., and Rigaud, A. Companionable: Graceful integration of mobile robot companion with a smart home environment. *Gerontechnology 8*, 3 (2009), 181.

[16] Baek, S.-M., Tachibana, D., Arai, F., and Fukuda, T. Situation based task selection mechanism for interactive robot system. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (2004), vol. 4, IEEE, pp. 3738–3743.

[17] Bailey, T., and Durrant-Whyte, H. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine 13*, 3 (2006), 108–117.

[18] Bareinboim, E., Forney, A., and Pearl, J. Bandits with unobserved confounders: A causal approach. In *Advances in Neural Information Processing Systems* (2015), pp. 1342–1350.

[19] Beetz, M., Mösenlechner, L., and Tenorth, M. Cram–a cognitive robot abstract machine for everyday manipulation in human environments. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on* (2010), IEEE, pp. 1012–1017.

[20] Ben Ghezala, M. W. *Compréhension dynamique du contexte pour l'aide à l'opérateur en robotique*. PhD thesis, Evry, Institut national des télécommunications, 2015.

[21] Bertolini, A., Salvini, P., Pagliai, T., Morachioli, A., Acerbi, G., Cavallo, F., Turchetti, G., and Dario, P. On robots and insurance. *International Journal of Social Robotics 8*, 3 (2016), 381–391.

[22] Bevilacqua, R., Felici, E., Marcellini, F., Glende, S., Klemcke, S., Conrad, I., Esposito, R., Cavallo, F., and Dario, P. Robot-era project: Preliminary results on the system usability. In *International Conference of Design, User Experience, and Usability* (2015), Springer, pp. 553–561.

[23] Blunsom, P. Hidden markov models. *Lecture notes, August 15* (2004), 18–19.

[24] Bobillo, F., Delgado, M., and Gómez-Romero, J. Crisp representations and reasoning for fuzzy ontologies. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 17*, 04 (2009), 501–530.

[25] Bobillo, F., and Straccia, U. Fuzzy ontology representation using owl 2. *International Journal of Approximate Reasoning 52*, 7 (2011), 1073–1094.

[26] Borja, R., De La Pinta, J. R., Álvarez, A., and Maestre, J. M. Integration of service robots in the smart home by means of upnp: A surveillance robot case study. *Robotics and Autonomous Systems 61*, 2 (2013), 153–160.

[27] Bouneffouf, D. *DRARS, A Dynamic Risk-Aware Recommender System*. PhD thesis, Institut National des Télécommunications, 2013.

[28] Bouneffouf, D., Bouzeghoub, A., and Ganarski, A. L. Risk-aware recommender systems. In *International Conference on Neural Information Processing* (2013), Springer, pp. 57–65.

[29] Brenna, L., Demers, A., Gehrke, J., Hong, M., Ossher, J., Panda, B., Riedewald, M., Thatte, M., and White, W. Cayuga: a high-performance event processing engine. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (2007), ACM, pp. 1100–1102.

[30] Broughton, G., Krajník, T., Fernández-Carmona, M., Cielniak, G., Bellotto, N., et al. Rfid-based object localisation with a mobile robot to assist the elderly with mild cognitive impairments. In *Intelligent Environments (Workshops)* (2016), pp. 366–375.

[31] Broxvall, M., Gritti, M., Saffiotti, A., Seo, B.-S., and Cho, Y.-J. Peis ecology: Integrating robots into smart environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* (2006), IEEE, pp. 212–218.

[32] Broxvall, M., Seo, B.-S., and Kwon, W. The peis kernel: A middleware for ubiquitous robotics. In *In Proc. of the IROS-07 Workshop on Ubiquitous Robotic Space Design and Applications* (2007), pp. 212–218.

[33] Burghart, C., Mikut, R., Stiefelhagen, R., Asfour, T., Holzapfel, H., Steinhaus, P., and Dillmann, R. A cognitive architecture for a humanoid robot: A first approach. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on* (2005), IEEE, pp. 357–362.

[34] Chibani, A., Amirat, Y., Mohammed, S., Matson, E., Hagita, N., and Barreto, M. Ubiquitous robotics: Recent challenges and future trends. *Robotics and Autonomous Systems 61*, 11 (2013), 1162–1172.

[35] Compton, M., Neuhaus, H., Taylor, K., and Parashar, A. Semantic sensor network ontology, 2013.

[36] Cooley, R., Mobasher, B., and Srivastava, J. Web mining: Information and pattern discovery on the world wide web. In *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on* (1997), IEEE, pp. 558–567.

[37] Cugola, G., Margara, A., Matteucci, M., and Tamburrelli, G. Introducing uncertainty in complex event processing: model, implementation, and validation. *Computing 97*, 2 (2015), 103–144.

[38] Currie, K., and Tate, A. O-plan: the open planning architecture. *Artificial intelligence 52*, 1 (1991), 49–86.

[39] Da Costa, P. C. G., Laskey, K. B., and Laskey, K. J. Pr-owl: A bayesian ontology language for the semantic web. In *Uncertainty Reasoning for the Semantic Web I*. Springer, 2008, pp. 88–107.

[40] Di Rocco, M., Pecora, F., Sivakumar, P. K., and Saffiotti, A. Configuration planning with multiple dynamic goals. In *AAAI Spring Symposium: Designing Intelligent Robots* (2013).

[41] Di Rocco, M., Sathyakeerthy, S., Grosinger, J., Pecora, F., Saffiotti, A., Bonaccorsi, M., Cavallo, F., Limosani, R., Manzi, A., Teti, G., et al. A planner for ambient assisted living: From high-level reasoning to low-level robot execution and back. In *2014 AAAI Spring Symposium Series* (2014).

[42] Ding, H., Heyn, J., Matthias, B., and Staab, H. Structured collaborative behavior of industrial robots in mixed human-robot environments. In *Automation Science and Engineering (CASE), 2013 IEEE International Conference on* (2013), IEEE, pp. 1101–1106.

[43] Du, Y., Wang, W., and Wang, L. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1110–1118.

[44] Durrant-Whyte, H., and Bailey, T. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine 13*, 2 (2006), 99–110.

[45] D'Aniello, G., Loia, V., and Orciuoli, F. A multi-agent fuzzy consensus model in a situation awareness framework. *Applied Soft Computing 30* (2015), 430–440.

[46] El-Yacoubi, M. A., He, H., Roualdes, F., Selmi, M., Hariz, M., and Gillet, F. Vision-based recognition of activities by a humanoid robot. *International Journal of Advanced Robotic Systems 12* (2015).

[47] Endsley, M. R. Toward a theory of situation awareness in dynamic systems. *Human factors 37*, 1 (1995), 32–64.

[48] Erol, K., Hendler, J., and Nau, D. S. Htn planning: Complexity and expressivity. In *AAAI* (1994), vol. 94, pp. 1123–1128.

[49] Erol, K., Hendler, J. A., and Nau, D. S. Umcp: A sound and complete procedure for hierarchical task-network planning. In *AIPS* (1994), vol. 94, pp. 249–254.

[50] event stream intelligence, E. Esper–complex event processing, 2010.

[51] Fikes, R. E., and Nilsson, N. J. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence 2*, 3-4 (1971), 189–208.

[52] Fitzgerald, C. Developing baxter. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on* (2013), IEEE, pp. 1–6.

[53] Friedman, N., Geiger, D., and Goldszmidt, M. Bayesian network classifiers. *Machine learning 29*, 2-3 (1997), 131–163.

[54] Garivier, A., and Moulines, E. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory* (2011), Springer, pp. 174–188.

[55] Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., and Tu, S. W. The evolution of protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies 58*, 1 (2003), 89–123.

[56] Ghezala, M. W. B. *Compréhension dynamique du contexte pour l'aide à l'opérateur en robotique*. PhD thesis, Institut National des Télécommunications, 2015.

[57] Ghezala, M. W. B., Bouzeghoub, A., and Leroux, C. Rsaw: A situation awareness system for autonomous robots. In *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on* (2014), IEEE, pp. 450–455.

[58] Glas, D. F., Kanda, T., Ishiguro, H., and Hagita, N. Simultaneous people tracking and localization for social robots using external laser range finders. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on* (2009), IEEE, pp. 846–853.

[59] Glas, D. F., Morales, Y., Kanda, T., Ishiguro, H., and Hagita, N. Simultaneous people tracking and robot localization in dynamic social spaces. *Autonomous Robots 39*, 1 (2015), 43–63.

[60] Glende, S., Conrad, I., Krezdorn, L., Klemcke, S., and Krätzel, C. Increasing the acceptance of assistive robots for older people through marketing strategies based on stakeholder needs. *International Journal of Social Robotics 8*, 3 (2016), 355–369.

[61] Gomes, P. F., Segura, E. M., Cramer, H., Paiva, T., Paiva, A., and Holmquist, L. E. Vipleo and phypleo: Artificial pet with two embodiments. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology* (2011), ACM, p. 3.

[62] Gori, I., Aggarwal, J., Matthies, L., and Ryoo, M. S. Multitype activity recognition in robot-centric scenarios. *IEEE Robotics and Automation Letters 1*, 1 (2016), 593–600.

[63] Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., and Maisonnier, B. Mechatronic design of nao humanoid. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (2009), IEEE, pp. 769–774.

[64] Graeser, A. Ambient intelligence and rehabilitation robots–a necessary symbiosis for robust operation in unstructured environments. In *Electronics and Telecommunications (ISETC), 2010 9th International Symposium on* (2010), IEEE, pp. 9–16.

[65] Gritti, M., Broxvall, M., and Saffiotti, A. Reactive self-configuration of an ecology of robots. In *Proc of the ICRA-07 Workshop on Network Robot Systems, Rome, Italy* (2007).

[66] Gross, H.-M., Schroeter, C., Mueller, S., Volkhardt, M., Einhorn, E., Bley, A., Langner, T., Martin, C., and Merten, M. I'll keep an eye on you: Home robot companion for elderly people with cognitive impairment. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on* (2011), IEEE, pp. 2481–2488.

[67] Gross, H.-M., Schroeter, C., Mueller, S., Volkhardt, M., Einhorn, E., Bley, A., Martin, C., Langner, T., and Merten, M. Progress in developing a socially assistive mobile home robot companion for the elderly with mild cognitive impairment. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on* (2011), IEEE, pp. 2430–2437.

[68] Guarino, D., and Saffiotti, A. Using fuzzy logic to monitor the state of an ubiquitous robotic system. *Journal of Uncertain Systems 2*, 2 (2008), 121–132.

[69] Hamid, R., Maddi, S., Bobick, A., and Essa, I. Unsupervised analysis of activity sequences using event-motifs. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks* (2006), ACM, pp. 71–78.

[70] Han, Y., Han, M., Lee, S., Sarkar, A., and Lee, Y.-K. A framework for supervising lifestyle diseases using long-term activity monitoring. *Sensors 12*, 5 (2012), 5363–5379.

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

[72] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. Support vector machines. *IEEE Intelligent Systems and their applications 13*, 4 (1998), 18–28.

[73] Hendrich, N., Bistry, H., and Zhang, J. Peis, mira, and ros: Three frameworks, one service robot–a tale of integration. In *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on* (2014), IEEE, pp. 1749–1756.

[74] Henricksen, K., and Indulska, J. Developing context-aware pervasive computing applications: Models and approach. *Pervasive and mobile computing 2*, 1 (2006), 37–64.

[75] Hoque, E., Dickerson, R. F., Preum, S. M., Hanson, M., Barth, A., and Stankovic, J. A. Holmes: A comprehensive anomaly detection system for daily in-home activities. In *Distributed Computing in Sensor Systems (DCOSS), 2015 International Conference on* (2015), IEEE, pp. 40–51.

[76] Hornyak, T. Meet pepper, the 'love-powered'humanoid robot that knows how you're feeling, 2014.

[77] Hugel, V., Amouroux, G., Costis, T., Bonnin, P., and Blazevic, P. Specifications and design of graphical interface for hierarchical finite state machines. In *Robot Soccer World Cup* (2005), Springer, pp. 648–655.

[78] Jakkula, V. R., and Cook, D. J. Detecting anomalous sensor events in smart home data for enhancing the living experience. *Artificial intelligence and smarter living 11*, 201 (2011), 1.

[79] Janjua, Z. H., Riboni, D., and Bettini, C. Towards automatic induction of abnormal behavioral patterns for recognizing mild cognitive impairment. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing* (2016), ACM, pp. 143–148.

[80] Jarraya, A., Ramoly, N., Bouzeghoub, A., Arour, K., Borgi, A., and Finance, B. Fscep: a new model for context perception in smart homes. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (2016), Springer, pp. 465–484.

[81] Jarraya, A., Ramoly, N., Bouzeghoub, A., Arour, K., Borgi, A., and Finance, B. A fuzzy semantic cep model for situation identification in smart homes. In *ECAI 2016: 22nd European Conference on Artificial Intelligence* (2016), vol. 285, IOS Press, pp. 1678–1679.

[82] Jokinen, K., and Wilcock, G. Multimodal open-domain conversations with the nao robot. In *Natural Interaction with Robots, Knowbots and Smartphones*. Springer, 2014, pp. 213–224.

[83] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence 101*, 1 (1998), 99–134.

[84] Kalyan, A., Gopalan, S., and Sridhar, V. Hybrid context model based on multilevel situation theory and ontology for contact centers. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on* (2005), IEEE, pp. 3–7.

[85] Kamei, K., Nishio, S., Hagita, N., and Sato, M. Cloud networked robotics. *IEEE Network 26*, 3 (2012).

[86] Kapotoglu, M., Koc, C., and Sariel, S. Robots avoid potential failures through experience-based probabilistic planning. In *Informatics in Control, Automation and Robotics (ICINCO), 2015 12th International Conference on* (2015), vol. 2, IEEE, pp. 111–120.

[87] Karlsson, L. Conditional progressive planning under uncertainty. In *IJCAI* (2001), pp. 431–438.

[88] Keßler, J., König, A., and Gross, H.-M. An improved sensor model on appearance based slam. *Autonome Mobile Systeme 2009* (2009), 153–160.

[89] Khaliq, A. A., Pecora, F., and Saffiotti, A. Inexpensive, reliable and localization-free navigation using an rfid floor. In *Mobile Robots (ECMR), 2015 European Conference on* (2015), IEEE, pp. 1–7.

[90] Khaliq, A. A., and Saffiotti, A. Stigmergy at work: Planning and navigation for a service robot on an rfid floor. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on* (2015), IEEE, pp. 1085–1092.

[91] Kim, J.-H., Kim, Y.-D., and Lee, K.-H. The third generation of robotics: Ubiquitous robot. In *Proc of the 2nd Int Conf on Autonomous Robots and Agents* (2004).

[92] Kim, J.-H., Lee, K.-H., Kim, Y.-D., Kuppuswamy, N. S., and Jo, J. Ubiquitous robot: A new paradigm for integrated services. In *Robotics and Automation, 2007 IEEE International Conference on* (2007), IEEE, pp. 2853–2858.

[93] Klir, G., and Yuan, B. *Fuzzy sets and fuzzy logic*, vol. 4. Prentice hall New Jersey, 1995.

[94] Knublauch, H., Fergerson, R. W., Noy, N. F., and Musen, M. A. The protégé owl plugin: An open development environment for semantic web applications. In *International Semantic Web Conference* (2004), Springer, pp. 229–243.

[95] Kwon, H. T., Yoon, W. C., Kim, R., and Kim, S. Building a hierarchical robot task from multiple task procedures. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on* (2011), IEEE, pp. 633–636.

[96] Kwon, W. Y., and Suh, I. H. Proactive planning using a hybrid temporal influence diagram for human assistive robots. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (2013), IEEE, pp. 1785–1791.

[97] Lafferty, J., McCallum, A., and Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

[98] Lallement, R., De Silva, L., and Alami, R. Hatp: An htn planner for robotics. *arXiv preprint arXiv:1405.5345* (2014).

[99] Laskey, K. B. Mebn: A language for first-order bayesian knowledge bases. *Artificial intelligence 172*, 2-3 (2008), 140–178.

[100] Lattimore, F., Lattimore, T., and Reid, M. D. Causal bandits: Learning good interventions via causal inference. In *Advances in Neural Information Processing Systems* (2016), pp. 1181–1189.

[101] LeBlanc, K., and Saffiotti, A. Cooperative anchoring in heterogeneous multi-robot systems. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (2008), IEEE, pp. 3308–3314.

[102] Lee, O.-J., and Jung, J. E. Sequence clustering-based automated rule generation for adaptive complex event processing. *Future Generation Computer Systems* (2016).

[103] Lee, Y.-S., and Cho, S.-B. Activity recognition using hierarchical hidden markov models on a smartphone with 3d accelerometer. In *International Conference on Hybrid Artificial Intelligence Systems* (2011), Springer, pp. 460–467.

[104] Lindstrom, M., Oreback, A., and Christensen, H. I. Berra: A research architecture for service robots. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on* (2000), vol. 4, IEEE, pp. 3278–3283.

[105] Liu, L., Stroulia, E., Nikolaidis, I., Miguel-Cruz, A., and Rincon, A. R. Smart homes and home health monitoring technologies for older adults: A systematic review. *International journal of medical informatics 91* (2016), 44–59.

[106] Lotfi, A., Langensiepen, C., Mahmoud, S. M., and Akhlaghinia, M. J. Smart homes for the elderly dementia sufferers: identification and prediction of abnormal behaviour. *Journal of ambient intelligence and humanized computing 3*, 3 (2012), 205–218.

[107] Luckham, D. *The power of events*, vol. 204. Addison-Wesley Reading, 2002.

[108] Lundh, R., Karlsson, L., and Saffiotti, A. Plan-based configuration of an ecology of robots. In *Robotics and Automation, 2007 IEEE International Conference on* (2007), IEEE, pp. 64–70.

[109] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA., pp. 281–297.

[110] Mahajan, A., and Teneketzis, D. Multi-armed bandit problems. *Foundations and Applications of Sensor Management* (2008), 121–151.

[111] Marzinotto, A., Colledanchise, M., Smith, C., and Ögren, P. Towards a unified behavior trees framework for robot control. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on* (2014), IEEE, pp. 5420–5427.

[112] McCarthy, J. Situation calculus with concurrent events and narrative.

[113] McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. Pddl-the planning domain definition language.

[114] Mckeever, S., Ye, J., Coyle, L., Bleakley, C., and Dobson, S. Activity recognition using temporal evidence theory. *Journal of Ambient Intelligence and Smart Environments 2*, 3 (2010), 253–269.

[115] Medjahed, H., Istrate, D., Boudy, J., and Dorizzi, B. A fuzzy logic system for home elderly people monitoring (emutem). *Fuzzy Systems* (2009).

[116] Milliez, G., Lallement, R., Fiore, M., and Alami, R. Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction* (2016), IEEE Press, pp. 43–50.

[117] Minnen, D., Starner, T., Essa, I., and Isbell, C. Discovering characteristic actions from on-body sensor data. In *Wearable computers, 2006 10th IEEE international symposium on* (2006), IEEE, pp. 11–18.

[118] Montreuil, V., Clodic, A., Ransan, M., and Alami, R. Planning human centered robot activities. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on* (2007), IEEE, pp. 2618–2623.

[119] Moore, D., and Essa, I. Recognizing multitasked activities from video using stochastic context-free grammar. In *AAAI/IAAI* (2002), pp. 770–776.

[120] Mouna Selmi, M. E.-Y., and Dorizzi, B. A two-layer discriminative model for human activity recognition. *IET Computer Vision* (2016).

[121] Murphy, K. P. Naive bayes classifiers. *University of British Columbia* (2006).

[122] Nassour, J., Hugel, V., Ouezdou, F. B., and Cheng, G. Qualitative adaptive reward learning with success failure maps: Applied to humanoid robot walking. *IEEE transactions on neural networks and learning systems 24*, 1 (2013), 81–93.

[123] Nau, D., Cao, Y., Lotem, A., and Munoz-Avila, H. Shop: Simple hierarchical ordered planner. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2* (1999), Morgan Kaufmann Publishers Inc., pp. 968–973.

[124] Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. Shop2: An htn planning system. *Journal of artificial intelligence research 20* (2003), 379–404.

[125] Novák, M., Bi·as, M., and Jakab, F. Unobtrusive anomaly detection in presence of elderly in a smart-home environment. In *ELEKTRO, 2012* (2012), IEEE, pp. 341–344.

[126] Novák, M., Jakab, F., and Lain, L. Anomaly detection in user daily patterns in smart-home environment. *J. Sel. Areas Health Inform 3* (2013), 1–11.

[127] Nunez, E., Matsuda, S., Hirokawa, M., and Suzuki, K. Humanoid robot assisted training for facial expressions recognition based on affective feedback. In *International Conference on Social Robotics* (2015), Springer, pp. 492–501.

[128] Okeyo, G., Chen, L., and Wang, H. Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes. *Future Generation Computer Systems 39* (2014), 29–43.

[129] Orkin, J. Agent architecture considerations for real-time planning in games. In *AIIDE* (2005), pp. 105–110.

[130] Ortega, P. A., and Braun, D. A. Generalized thompson sampling for sequential decision-making and causal inference. *Complex Adaptive Systems Modeling 2*, 1 (2014), 2.

[131] Ouali, L. O., Rich, C., and Sabouret, N. Plan recovery in reactive htns using symbolic planning. In *International Conference on Artificial General Intelligence* (2015), Springer, pp. 320–330.

[132] Parekh, H. S., Thakore, D. G., and Jaliya, U. K. A survey on object detection and tracking methods. *International Journal of Innovative Research in Computer and Communication Engineering 2*, 2 (2014), 2970–2979.

[133] Park, K.-H., Lee, H.-E., Kim, Y., and Bien, Z. Z. A steward robot for human-friendly human-machine interaction in a smart house environment. *IEEE Transactions on Automation Science and Engineering 5*, 1 (2008), 21–25.

[134] Pearl, J. *Causality*. Cambridge university press, 2009.

[135] Pednault, E. P. Formulating multiagent, dynamic-world problems in the classical planning framework. *Reasoning about actions and plans* (1986), 47–82.

[136] Peshkin, M. A., Colgate, J. E., Wannasuphoprasit, W., Moore, C. A., Gillespie, R. B., and Akella, P. Cobot architecture. *IEEE Transactions on Robotics and Automation 17*, 4 (2001), 377–390.

[137] Pomerleau, D. A. Knowledge-based training of artificial neural networks for autonomous robot driving. *Robot learning* (1993), 19–43.

[138] Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., and De Bosschere, K. Towards an extensible context ontology for ambient intelligence. In *EUSAI* (2004), vol. 3295, Springer, pp. 148–159.

[139] Ramoly, N., Bouzeghoub, A., and Finance, B. A framework for service robots in smart home: an efficient solution for domestic healthcare. In *IRBM - Innovation and Research in BioMedical Engineering*.

[140] Ramoly, N., Bouzeghoub, A., and Finance, B. Context-aware planning by refinement for personal robots in smart homes. In *ISR 2016: 47st International Symposium on Robotics; Proceedings of* (2016), VDE, pp. 1–8.

[141] Ramoly, N., Bouzeghoub, A., and Finance, B. A causal multi-armed bandit approach for domestic robots' failure avoidance. In *International Conference on Neural Information Processing* (2017), Springer, pp. 90–99.

[142] Ramoly, N., Bouzeghoub, A., and Finance, B. A framework for service robots in smart home: an efficient solution for domestic healthcare. In *Journées d'Etude sur la TéléSANté, 6ème edition* (2017).

[143] Ramoly, N., Sfar, H., Bouzeghoub, A., and Finance, B. Leaf: Using semantic based experience to prevent task failures. In *Field and Service Robotics* (2017), Springer, pp. 681–697.

[144] Ramoly, N., Vassout, V., Bouzeghoub, A., El Yacoubi, M. A., and Hariz, M. Refining visual activity recognition with semantic reasoning. In *Advanced Information Networking and Applications (AINA), 2017 IEEE 31st International Conference on* (2017), IEEE, pp. 720–727.

[145] Ramos, C., Augusto, J. C., and Shapiro, D. Ambient intelligence–the next step for artificial intelligence. *IEEE Intelligent Systems 23*, 2 (2008), 15–18.

[146] Riboni, D., Bettini, C., Civitarese, G., Janjua, Z. H., and Helaoui, R. Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment. In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on* (2015), IEEE, pp. 149–154.

[147] Riboni, D., Bettini, C., Civitarese, G., Janjua, Z. H., and Helaoui, R. Smartfaber: Recognizing fine-grained abnormal behaviors for early detection of mild cognitive impairment. *Artificial intelligence in medicine 67* (2016), 57–74.

[148] Riboni, D., Pareschi, L., Radaelli, L., and Bettini, C. Is ontology-based activity recognition really effective? In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on* (2011), IEEE, pp. 427–431.

[149] Richardson, M., and Domingos, P. Markov logic networks. *Machine learning 62*, 1 (2006), 107–136.

[150] Rockel, S., Neumann, B., Zhang, J., Dubba, K. S. R., Cohn, A. G., Konecny, S., Mansouri, M., Pecora, F., Saffiotti, A., Günther, M., et al. An ontology-based multi-level robot architecture for learning from experiences. In *AAAI Spring Symposium: Designing Intelligent Robots* (2013), vol. 2, pp. 2–4.

[151] Rodríguez, N. D., Cuéllar, M. P., Lilius, J., and Calvo-Flores, M. D. A fuzzy ontology for semantic modelling and recognition of human behaviour. *Knowledge-Based Systems 66* (2014), 46–60.

[152] Rougui, J., Istrate, D., and Souidene, W. Audio sound event identification for distress situations and context awareness. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE* (2009), IEEE, pp. 3501–3504.

[153] Sacerdoti, E. D. Planning in a hierarchy of abstraction spaces. *Artificial intelligence 5*, 2 (1974), 115–135.

[154] Safavian, S. R., and Landgrebe, D. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics 21*, 3 (1991), 660–674.

[155] Saffiotti, A., and Broxvall, M. Peis ecologies: Ambient intelligence meets autonomous robotics. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies* (2005), ACM, pp. 277–281.

[156] Saffiotti, A., Broxvall, M., Gritti, M., LeBlanc, K., Lundh, R., Rashid, J., Seo, B., and Cho, Y.-J. The peis-ecology project: vision and results. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (2008), IEEE, pp. 2329–2335.

[157] Sariel, S., Yildiz, P., Karapinar, S., Altan, D., and Kapotoglu, M. Robust task execution through experience-based guidance for cognitive robots. In *Advanced Robotics (ICAR), 2015 International Conference on* (2015), IEEE, pp. 663–668.

[158] Selmi, M. *Reconnaissance d'activités humaines à partir de séquences vidéo. (Human activity recognition from video sequences)*. PhD thesis, Telecom & Management SudParis, Évry, Essonne, France, 2014.

[159] Sen, R., Shanmugam, K., Kocaoglu, M., Dimakis, A. G., and Shakkottai, S. Contextual bandits with latent confounders: An nmf approach. *arXiv preprint arXiv:1606.00119* (2016).

[160] Sfar, H., Bouzeghoub, A., and Ramoly, N. Activity recognition for anomalous situation detection. In *Journées d'Etude sur la TéléSANté, 6ème edition* (2017).

[161] Sfar, H., Bouzeghoub, A., Ramoly, N., and Boudy, J. Agacy monitoring: a hybrid model for activity recognition and uncertainty handling. In *European Semantic Web Conference* (2017), Springer, pp. 254–269.

[162] Sfar, H., Bouzeghoub, A., Ramoly, N., and Boudy, J. A novel hybrid model for activity recognition. In *Modeling Decisions for Artificial Intelligence* (2017), Springer, pp. 170–182.

[163] Sfar, H., Ramoly, N., Bouzeghoub, A., and Finance, B. Caredas: Context and activity recognition enabling detection of anomalous situation. In *Conference on Artificial Intelligence in Medicine in Europe* (2017), Springer, pp. 24–36.

[164] Shafer, G., et al. *A mathematical theory of evidence*, vol. 1. Princeton university press Princeton, 1976.

[165] Shamsuddin, S., Yussof, H., Ismail, L. I., Mohamed, S., Hanapiah, F. A., and Zahari, N. I. Initial response in hri-a case study on evaluation of child with autism spectrum disorders interacting with a humanoid robot nao. *Procedia Engineering 41* (2012), 1448–1455.

[166] Shiomi, M., Kanda, T., Ishiguro, H., and Hagita, N. Interactive humanoid robots for a science museum. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction* (2006), ACM, pp. 305–312.

[167] Simonnet, T., Couet, A., Ezvan, P., Givernaud, O., and Hillereau, P. Telemedicine platform enhanced visiophony solution to operate a robot-companion. *Advanced Techniques in Computing Sciences and Software Engineering* (2010), 301–305.

[168] Song, B., Tian, G., Li, G., Zhou, F., and Liu, D. Zigbee based wireless sensor networks for service robot intelligent space. In *Information Science and Technology (ICIST), 2011 International Conference on* (2011), IEEE, pp. 834–838.

[169] Stojkoska, B. L. R., and Trivodaliev, K. V. A review of internet of things for smart home: Challenges and solutions. *Journal of Cleaner Production 140* (2017), 1454–1464.

[170] Tate, A. *Project planning using a hierarchic non-linear planner*. Department of Artificial Intelligence, University of Edinburgh, 1976.

[171] Tenorth, M., and Beetz, M. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research 32*, 5 (2013), 566–590.

[172] Tesoriero, R., Tebar, R., Gallud, J. A., Lozano, M. D., and Penichet, V. M. R. Improving location awareness in indoor spaces using rfid technology. *Expert Systems with Applications 37*, 1 (2010), 894–898.

[173] Tian, W., and Geng, Y. A new household security robot system based on wireless sensor network. In *Future Information Technology and Management Engineering, 2009. FITME'09. Second International Conference on* (2009), IEEE, pp. 187–190.

[174] Vail, D. L., Veloso, M. M., and Lafferty, J. D. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems* (2007), ACM, p. 235.

[175] van Kasteren, T., and Krose, B. Bayesian activity recognition in residence for elders.

[176] Vidal, V., and Geffner, H. Branching and pruning: An optimal temporal pocl planner based on constraint programming. *Artificial Intelligence 170*, 3 (2006), 298–335.

[177] Volkhardt, M., Müller, S., Schröter, C., and Gross, H.-M. Real-time activity recognition on a mobile companion robot. In *55th Int. Scientific Colloquium* (2010).

[178] Volosyak, I., Ivlev, O., and Graser, A. Rehabilitation robot friend ii-the general concept and current implementation. In *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on* (2005), IEEE, pp. 540–544.

[179] Waldhart, J., Gharbi, M., and Alami, R. A novel software combining task and motion planning for human-robot interaction. In *AAAI Fall Symposia* (2016), p. 139.

[180] Wang, F., Liu, S., Liu, P., and Bai, Y. Bridging physical and virtual worlds: complex event processing for rfid data streams. In *Advances in Database Technology-EDBT 2006*. Springer, 2006, pp. 588–607.

[181] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 3169–3176.

[182] Wang, X. H., Zhang, D. Q., Gu, T., and Pung, H. K. Ontology based context modeling and reasoning using owl. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on* (2004), Ieee, pp. 18–22.

[183] Warfield, I., Hogg, C., Lee-Urban, S., and Munoz-Avila, H. Adaptation of hierarchical task network plans. In *FLAIRS conference* (2007), pp. 429–434.

[184] Wasserkrug, S., Gal, A., and Etzion, O. A model for reasoning with uncertain rules in event composition systems. *arXiv preprint arXiv:1207.1427* (2012).

[185] Weser, M., Off, D., and Zhang, J. Htn robot planning in partially observable dynamic environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on* (2010), IEEE, pp. 1505–1510.

[186] Wisspeintner, T., Van Der Zant, T., Iocchi, L., and Schiffer, S. Robocup@ home: Scientific competition and benchmarking for domestic service robots. *Interaction Studies 10*, 3 (2009), 392–426.

[187] Wu, E., Diao, Y., and Rizvi, S. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data* (2006), ACM, pp. 407–418.

[188] Wyatt, D., Philipose, M., and Choudhury, T. Unsupervised activity recognition using automatically mined common sense. In *AAAI* (2005), vol. 5, pp. 21–27.

[189] Ye, J., Dasiopoulou, S., Stevenson, G., Meditskos, G., Kontopoulos, E., Kompatsiaris, I., and Dobson, S. Semantic web technologies in pervasive computing: A survey and research roadmap. *Pervasive and Mobile Computing 23* (2015), 1–25.

[190] Ye, J., Dobson, S., and McKeever, S. Situation identification techniques in pervasive computing: A review. *Pervasive and mobile computing 8*, 1 (2012), 36–66.

[191] Zhang, H., Diao, Y., and Immerman, N. On complexity and optimization of expensive queries in complex event processing. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (2014), ACM, pp. 217–228.

[192] Zhang, S., Sridharan, M., and Wyatt, J. L. Mixed logical inference and probabilistic planning for robots in unreliable worlds. *IEEE Transactions on Robotics 31*, 3 (2015), 699–713.

[193] Zhou, Q., Simmhan, Y., and Prasanna, V. Scepter: Semantic complex event processing over end-to-end data flows. Tech. rep., Technical Report 12-926, Computer Science Department, University of Southern California, 2012.

Appendix A

VARSeR Experiments

In this appendix, we review our extensive experiments of our hybrid approach for activity recognition: VARSeR. In these experiments, we first evaluate the performance of the vision based recognition by itself, then we evaluate how our contribution improves the activity recognition.

A.1 Implementation

The vision based activity recognition by El-Yacoubi et al. [46] was already implemented. This approach was embedded on a Nao robot. A small behaviour was implemented on the robot to acquire a video from its camera and apply the algorithm on it.

Then, this process was altered to take into account context data provided by sensors from HadapTIC. Note that we used the HadapTIC as in 2014 and 2015, as the platform moved and was enriched in 2016. We used a server for gathering context data from Legrand devices, motion sensors and opening sensors, through ZigBee. We used another server to maintain the ontological context knowledge that is filled from the other server and mobile phones. Nao and this server communicate to perform the refinement: the robot provides its recognized activity, the server stores them in the ontology, performs its process and gives back its corrected results so that Nao can vocally say the detected activity. Note that the context knowledge server could have been embedded in the robot, but would have required a lot of engineering efforts.

A.2 Protocol

Experiments were conducted as follows. The robot was installed in the room, in most cases it was sitting (as in Figure A.1) and facing the scene where the action was performed. Please note that for these experiments, the robot was not moving, in fact the protocol can be applied on a legless T14 Nao [1]. Each volunteer was asked to come in front of the robot and to do a gesture matching one scenario. The participants were asked to make a gesture for each of these three actions :

1. Make a phone call

[1] `http://doc.aldebaran.com/2-1/family/nao_t14/index_t14.html`

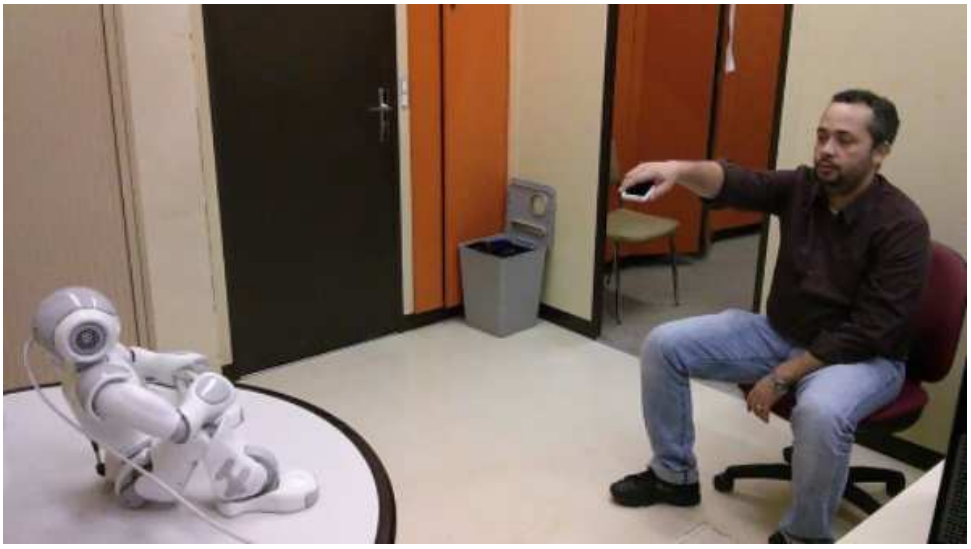


Figure A.1: Photo of our experiments for a "remote control" scenario

2. Activate the remote control

3. Open the door

One test iteration consists of the following protocol:

1. The Nao opens the process by telling when the process starts and when the video acquisition starts.

2. Once Nao announced the beginning of the record, the user does the gesture. A two seconds video of the action is captured by the robot. An example of a recording can be found in Figure 8.9.

3. Nao applies its own activity recognition algorithm. This step can take up to 15 seconds due to the heavy process of the visual recognition. Once Nao finishes, it sends its result to the server.

4. The server stores the result and queries the sensors in order to refine the activity recognition.

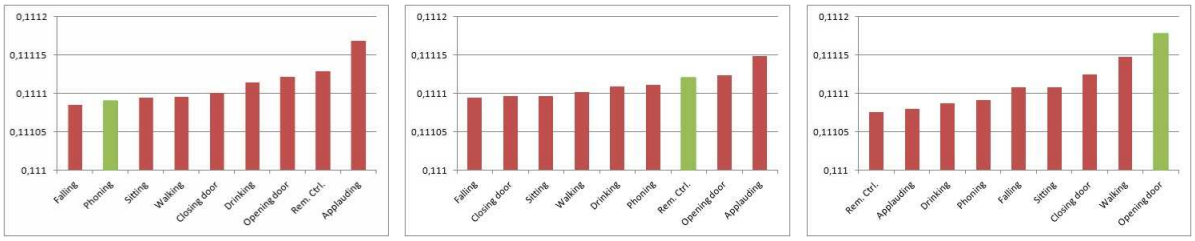
5. Once all data are collected, the server computes a new activity distribution and provides it to Nao.

6. Nao announces the identified activity, with and without refinement. All results are stored in the server to be recovered for analysis (see next Section).

You can find example of videos online[2]

The process is repeated 10 times per scenario and per volunteer. In other words, each volunteer participates in $3 * 10$ instances. On the overall, 120 records were obtained for each gesture. In every case, the results were analysed and are reviewed, as addressed in the next Section.

[2] `http://nara.wp.tem-tsp.eu/vision-activity-recognition/`



(a) Phoning (Scenario 1) (b) Remote Controlling (Scenario 2) (c) Opening door (Scenario 3)

Figure A.2: Average resulting distribution **without refinement**

The green column matches the actual observed action.

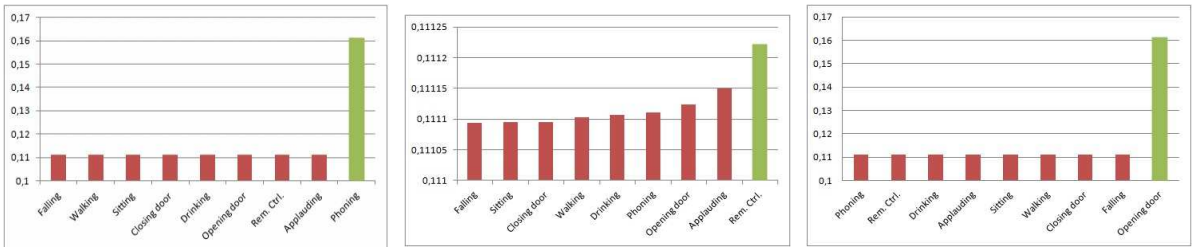
A.3 Results

The experiments helped to understand the performance of the vision-based recognition algorithm and the impact of our approach.

Firstly, let us have a look at the result of the activity recognition without refinement. Figure A.2 depicts the average activity distribution, provided by the robot without refinement for one given scenario. The charts show the ordered probability for each activity. The activity to recognize, matching the user actual action, is associated with the green column. Firstly, it is noticeable that the distribution is very tight: provided probabilities are close to each other. This result points out the difficulty for the vision-based algorithm to have a clear and strong classification by itself. Nevertheless, differences are observable to a certain scale and they provide an order between activities. Having said that, it is important to note that it may not be enough. Figure A.4a shows the successful recognition rates: with 6.40% for the "remote controlling" scenario, 53.97% for the "opening door" and 6.35% for the "phoning" one. The results are quite poor, but have to be put in the perspective. Results are varying according to the scenario. The "opening door" scenario provides the best results. As we can see in Figure A.2c, in most cases the robot identifies properly the activity, but still makes confusion in one out of two cases. With only 6.40% successful recognition rates, the robot seems to have difficulty to recognize the "remote controlling" activity. However, in Figure A.2b, in average, the proper activity is recognized in third place, behind the "opening door" and "applauding" activities. This is clearly due to a **gesture confusion**: all top three activities are illustrated by the rise of an arm and thus the difference can be tricky to distinguish. The phone scenario has very poor results. As we can see in Figure A.2a, it is almost the last activity in the distribution while being the one to be observed.

From these results, we can see the unreliability of the vision-based recognition in some cases. In fact, even if it achieves good results on datasets [46], it proves to be limited in more general scenarios. The process itself can be inaccurate, and is sensitive to confusion. This can be explained by the lack of training and this emphasize **the importance and the difficulty of learning** of a proper vision-based activity recognition algorithm. Through these results, we can clearly see the weaknesses of this approach and why it should be improved. We will now study the results obtained after applying the refinement process.

Figure A.3 shows the activity distribution provided by VARSeR after refinement with further context data. As we can see, for each scenario, after adjustment the observed activity is on average the



(a) Phoning (Scenario 1) (b) Remote Controlling (Scenario 2) (c) Opening door (Scenario 3)

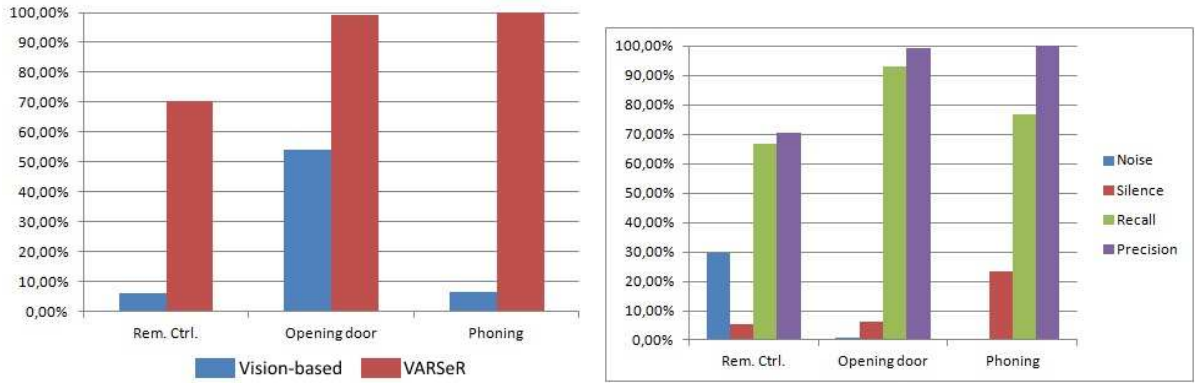
Figure A.3: Average resulting distribution **after refinement**

The green column matches the actual observed action.

most probable one, but with a much higher probability. This difference can be explained in two ways. Firstly, as expressed in the previous paragraph, raw activity distribution is very tight: every adjustment on a different scale can lead to such differences. In a more complete scenario, we can imagine a more disparate distribution due to the impact of multiple sensors at the same time, but in such a case, the role of the vision-based recognition seems weak. The second explanation is the "semantic" carried by the sensors. In fact, sensors themselves can give strong hint about what is going on. For example, the opening door sensors imply themselves almost certainly that the door is open. Thus, it greatly impacts the probability (Figure A.3c). The same goes for the "phoning" activity, as we can notice in Figure A.3a. The correlation between sensors and activities explains the gap of probability. Moreover, in the "remote controlling" scenario (Figure A.3b), the motion sensor has less impact as it is not specifically linked to this activity. Concerning the "opening door" and the "remote controlling" scenario, the refinement allows to **confirm the identified activity and/or to remove an ambiguity**. However, as for the "phoning" scenario, while the vision provided a very poor activity distribution, the refined result is actually correct. Thus, in this case, we can assert that the activity recognition was actually allowed solely through the context knowledge as the vision process did not provide any clue toward the right activity. In other words, in this case, the vision process outcome has no influence and could even be ignored. This **finding raises the question of the relevance of using vision-based algorithm** when other solutions are available. Putting aside the gap between refined and non-refined probabilities, the results show that **VARSeR does allow a correct activity recognition**, as sumarized in Figure A.4a.

Figure A.4a shows the activity recognition success rate with and without refinement. As previously explained, raw vision-based activity recognition had mediocre results while The refined "remote controlling" scenario has 71% successful recognition rate. The two others were almost reaching 100% of recognition with VARSeR. This is again explained by the impact of the sensors. These results have also to be put into perspective: in a more complex scenario with more activities and data sources, these rates would not be equally good. In this experiment, we aimed to test the refinement facing one single action with a few sensors. In a real application with much more context data that can be conflicting, confusion would be more present. Nevertheless, it shows the gain we can have by combining and/or relying on smart sensors.

Figure A.4b presents the noise, silence, recall and precision of the refinement process. These



(a) Successful activity recognition rate per scenario. Blue: without refinement. Red: with refinement

(b) Noise, Silence, Recall and Precision per scenario achieved by VARSeR

Figure A.4: VARSeR overall results

notions are defined as follows:

- **Noise:** Number of incorrect results provided, divided by the total number of results.
- **Silence:** Number of correct results not provided, divided by the total number of existing correct results. In our case, it is the proportion of distribution that were not refined due to missing data.
- **Recall:** Number of correct results provided, divided by the total number of existing correct results. It can be seen as the success rate considering the case the correction was not applied.
- **Precision:** Number of correct results provided, divided by the total number of results. It is almost similar to the success rate depicted in Figure A.4a.

Although correction is efficient, there were some instances where it has not been applied properly, in particular for the "remote controlling" and "phoning" scenarios, which have respectively a recall of only 67% and 77%. In fact, we observed that the sensors sometimes do not detect the action: even if they are reliable for activity recognition, they are not flawless. This underlines the problem of **sensors uncertainty** that is tackled by FSCEP.

A.4 Conclusion

To summarize these experiments, we can assert that combining vision and sensors from the environment allows to provide more reliable results. But, our main conclusion is that using both approaches at the same time is not always the best solution. In some cases, the vision process could be avoided, preventing huge costs in time and resources. In fact, we believe that **selecting a method to recognize activities** instead of combining them in any case would be a better solution. Typically in this case, the robot could select either the vision, the smart environment, or both of them, depending on the current case. This is obviously a challenging, yet promising, issue to tackle.

Appendix B

FAIRIE Detailed Implementation

In this appendix, we provide more details on FAIRIE implementation. We aim to show how modules communicate together within the framework. Figure B.3 shows the whole FAIRIE framework with all presented modules and communications between them. The related legend is presented in separated Figure B.2.

Figure B.3 represents the life of data from sensors to robot actions. It noticeably shows the different knowledges used, from both learning and specification: hence, it explains where and how knowledge is to be provided. Multiple types of communications are displayed. First, internal communications show the exchange within one module, these are mainly data transmission between process steps. Note that the video stream is directly provided to VARSeR as it is embedded in the robot. The second and most used communications are network exchanges through the UDP protocol. It was selected as it does not require maintained and continuous connection between endpoints, easing the interoperability and configuration between modules. Yet, other vector of communication could be used, including dedicated middleware. Typically, ROS can be used to handle all communications,: doing so is a perspective of improvement. Nevertheless, ROS was also used, in particular for handling DHTN and robot actions. Hence, ROS middleware features are the third type of communication. ROS publish/subscribe approach was widely used around DHTN, Figure B.4 shows the nodes and topics used within FAIRIE. We also used the notion of services of ROS, in particular for robot actions. As mentioned, each action of the robot is represented as a script carrying a ROS service that is then called by an operator manager.

With the current proposed modules, FAIRIE receives as inputs video streams and events. Events are previously formatted before being transmitted to FSCEP. This preliminary step is performed through dedicated interfaces or through a middleware. In our experiments, m4iot middleware was used to acquire data from sensors within the HadapTIC platform. m4iot relies on a network of *collectors*, that acquire and format raw data from sensors, *capsules*, that enrich those data, and *applications*, that use collected and generated data. Brokers are used to ensure communications with local and distant networks within the IoT. As depicted in Figure B.1, FAIRIE was integrated through a dedicated capsule. It transforms the data acquired by collectors to events, by adding the time notion, and provided to FAIRIE. Although not performed yet, having FAIRIE re-inject its computed knowledge, such as user activities or anomalous situations, in m4iot is a perspective: this motivated the creation of the dedicated *capsule* rather than an *application*.

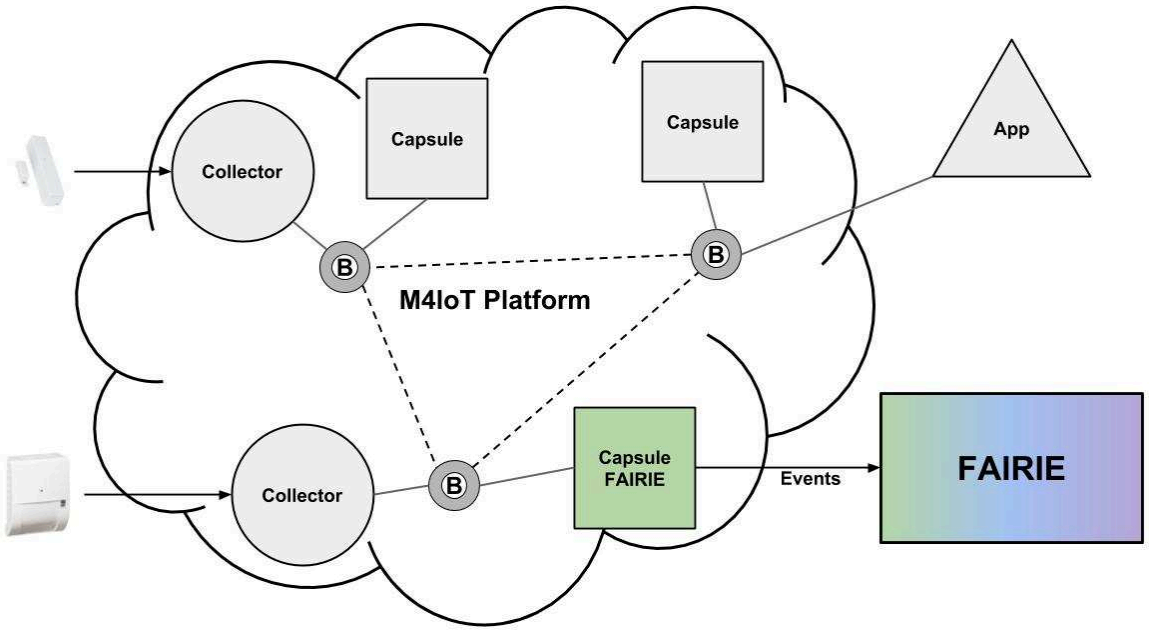


Figure B.1: Integration of FAIRIE with m4iot

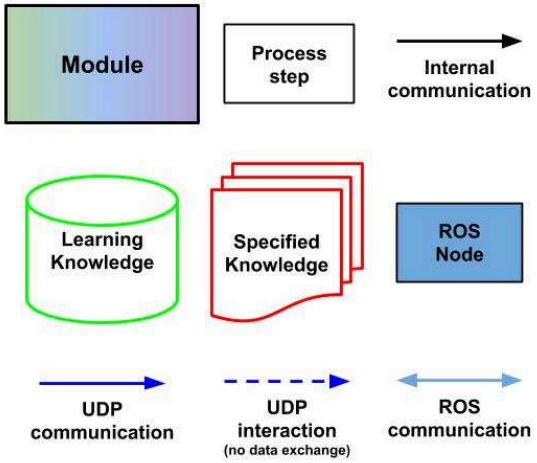


Figure B.2: Legend of Figure B.3

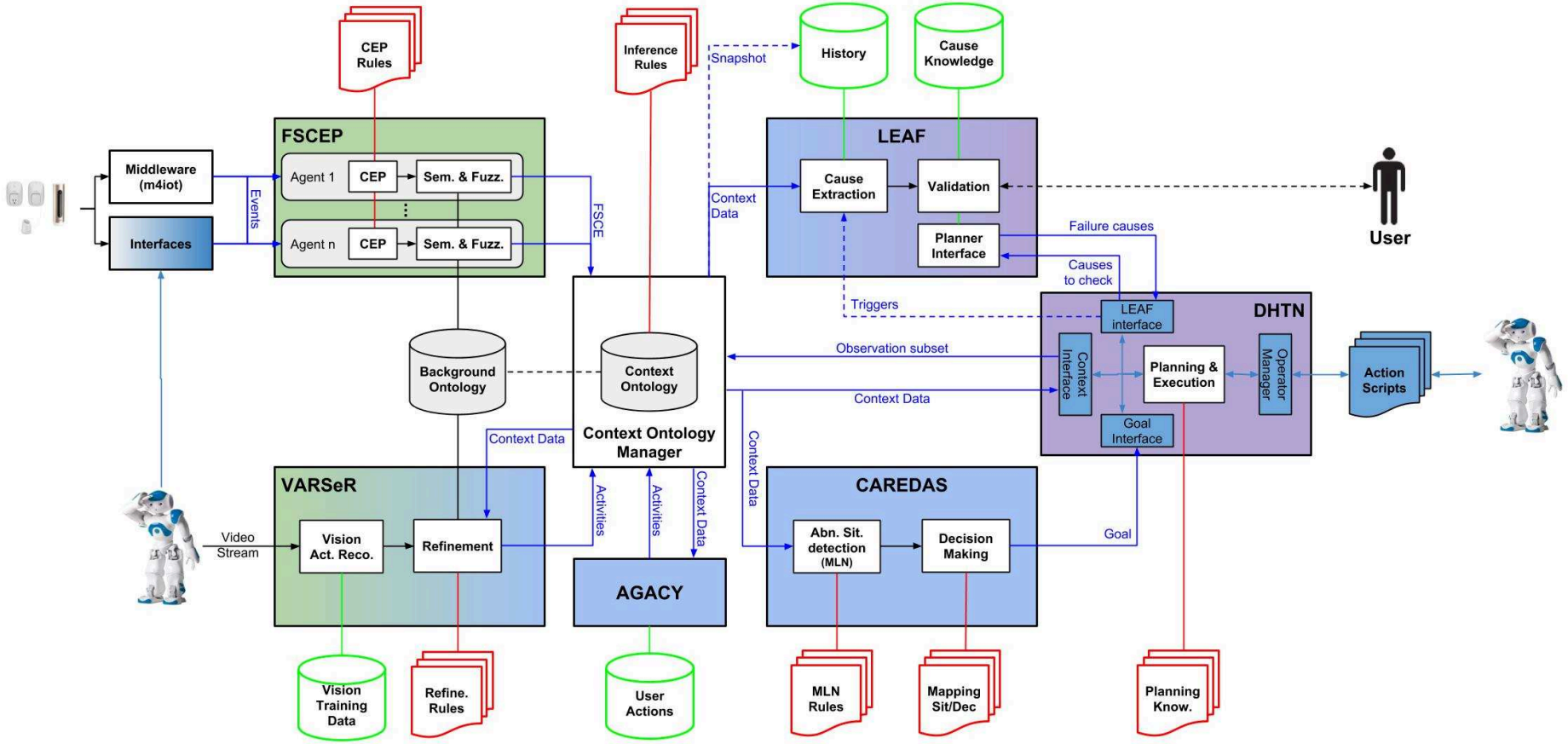


Figure B.3: FAIRIE implementation detailed architecture

Legend can be found in Figure B.2

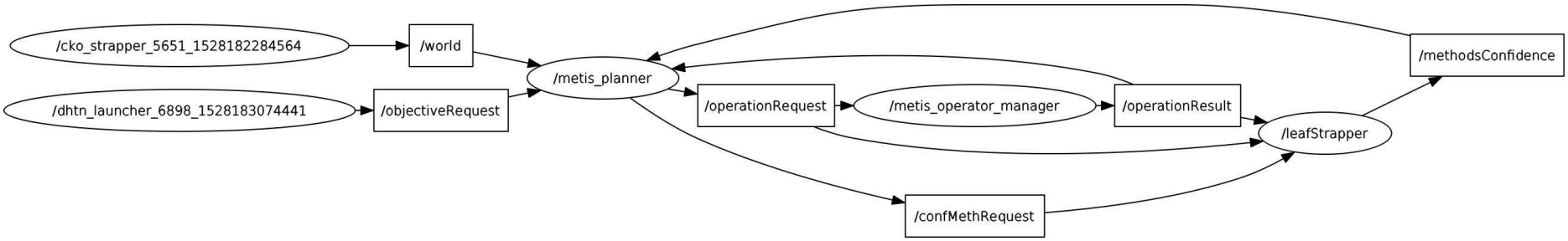


Figure B.4: ROS nodes and topics used

This graph is generated within ROS through *rqt_graph*. Circles are nodes, Rectangle are topics. Services are not represented. Note that *metis* is the former codename of DHTN.

cko_strapper is the node that acquires data from the context ontology. *dhtn_launcher* is in charge of gathering goals and starting plans. *metis_planner* is the DHTN planner itself. *metis_operator_manager* is the part of DHTN that launches operator and retrieves outcomes. *leafStrapper* focuses on interactions with LEAF.

Appendix C

A Nao Action Script

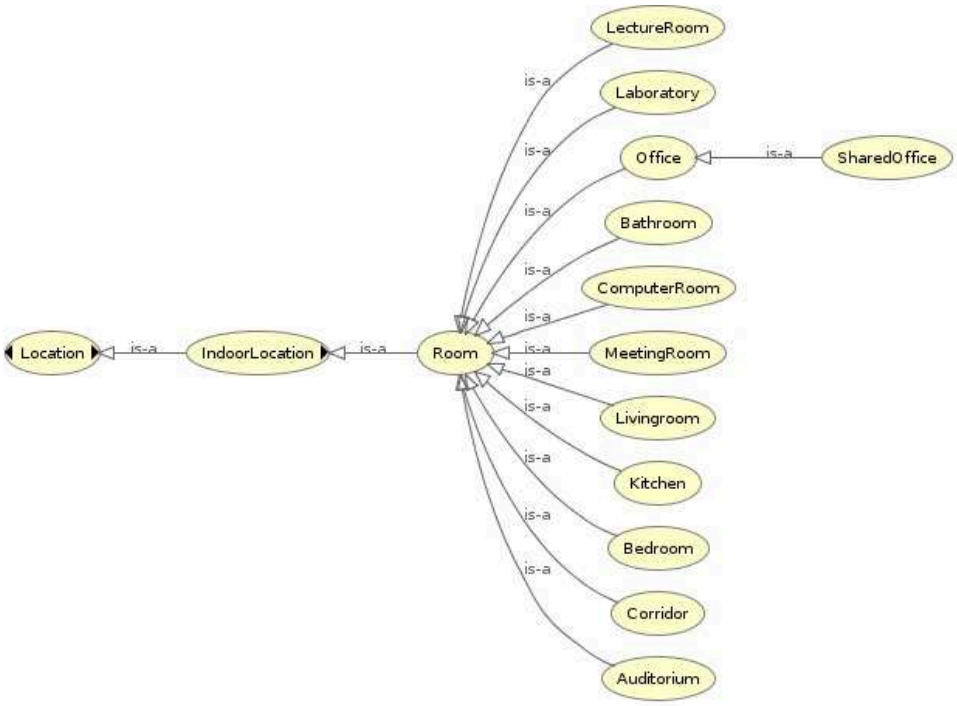
```
#!/usr/bin/env python

import rospy
import time
import math

from nao_action.srv import *

from naoqi import ALProxy


NAO_IP = '192.168.0.102'
NAO_PORT = 9559


#Perform the action consisting of asking the user to take his/her
#medicine and wait for his/her feedback.
def run_action(req):
        print "Running service"

        #Starting NaoQi modules
        tts  = ALProxy("ALTextToSpeech", NAO_IP, NAO_PORT)
        asr  = ALProxy("ALSpeechRecognition", NAO_IP, NAO_PORT)
        mem  = ALProxy("ALMemory", NAO_IP, NAO_PORT)

        #Reminding message
        tts.say("It is time for you to get your medecine, don't forget it.")

        #Getting the speech recognition ready
        asr.setLanguage("English")
        voc = ["ok"] #Recognizing word "ok"
        asr.setVocabulary(voc, False)

        #Starting speech recognition
        asr.subscribe("ASR")
```

```
        #Waiting for an answer
        timeOut = True #Set at false when exiting
        tries = 25
        words = {}

        for i in range(0, tries):
                time.sleep(0.5)
                print mem.getDataList("ord")
                if len( mem.getDataList("WordRecognized") ) > 0:
                        #Gathering recognized words
                        words = mem.getData("WordRecognized")
                        #Checking if at least one 'ok' was recognized
                        if len(words[0]) > 0:
                                timeOut = False
                                break

        #Ending speech recognision
        asr.unsubscribe("ASR")

        #If no answer: failures
        if timeOut :
                return "FAILURE"
        #If feedback, action is a success
        else:
                tts.say("Thank␣you␣!")
                return "SUCCESS"


#Creation of ROS node and service
def server():
        rospy.init_node('nao_vocal_alert_server')
        s = rospy.Service('nao_vocal_alert', action, run_action)
        print "Ready␣to␣run␣nao_vocal_alert"
        rospy.spin()


if __name__ == "__main__":
        server()
```

Appendix D

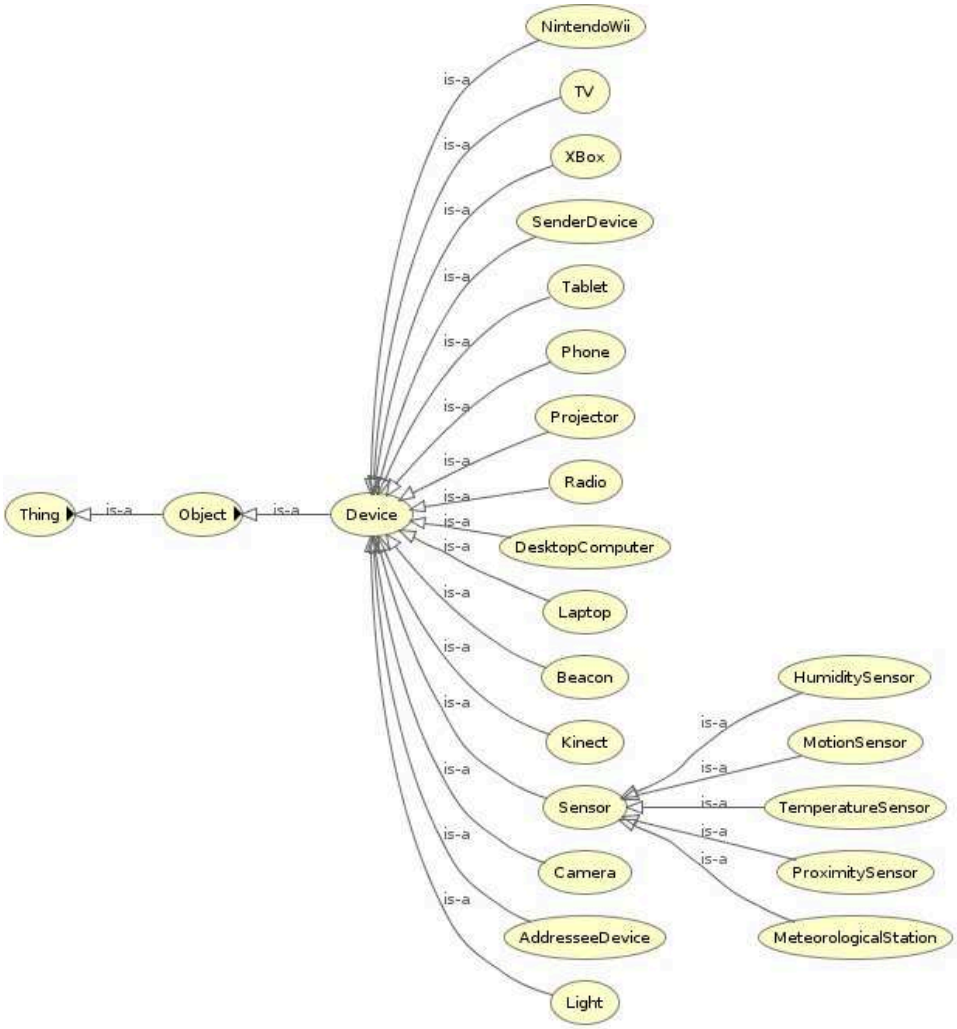
Ontology Structure

In this appendix, we present a part of the structure of the background and context ontologies we mainly used. It is based on the ontology proposed by Rodriguez et al. for human activity representation [1] [151]. Here, we focus on three key concepts that are locations, devices and activities. Note that all these figures were obtained from Protégé.

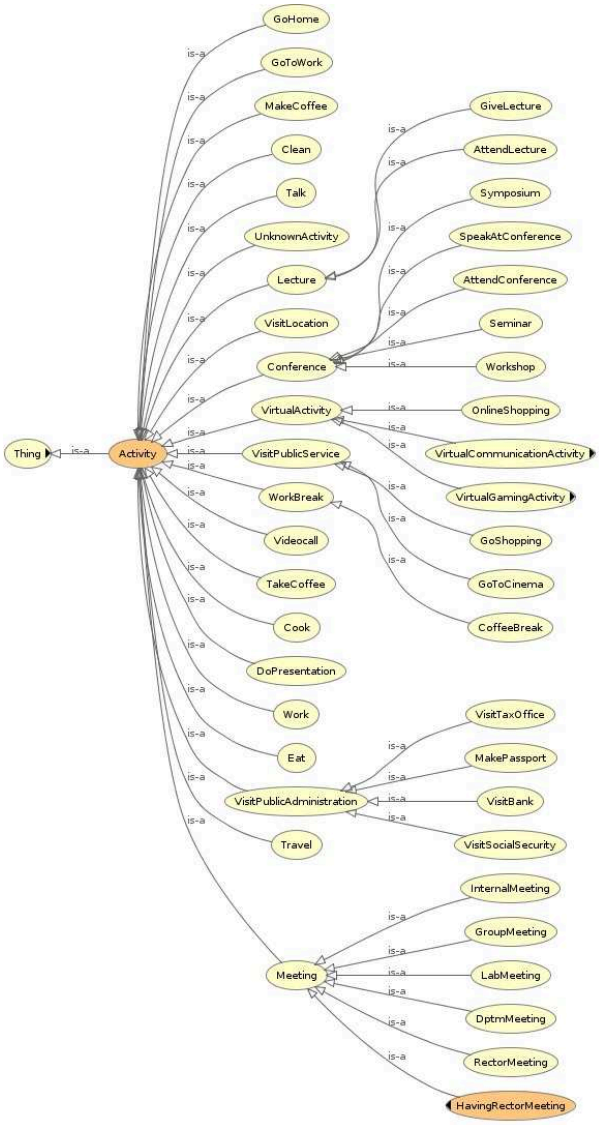


Classes related to locations

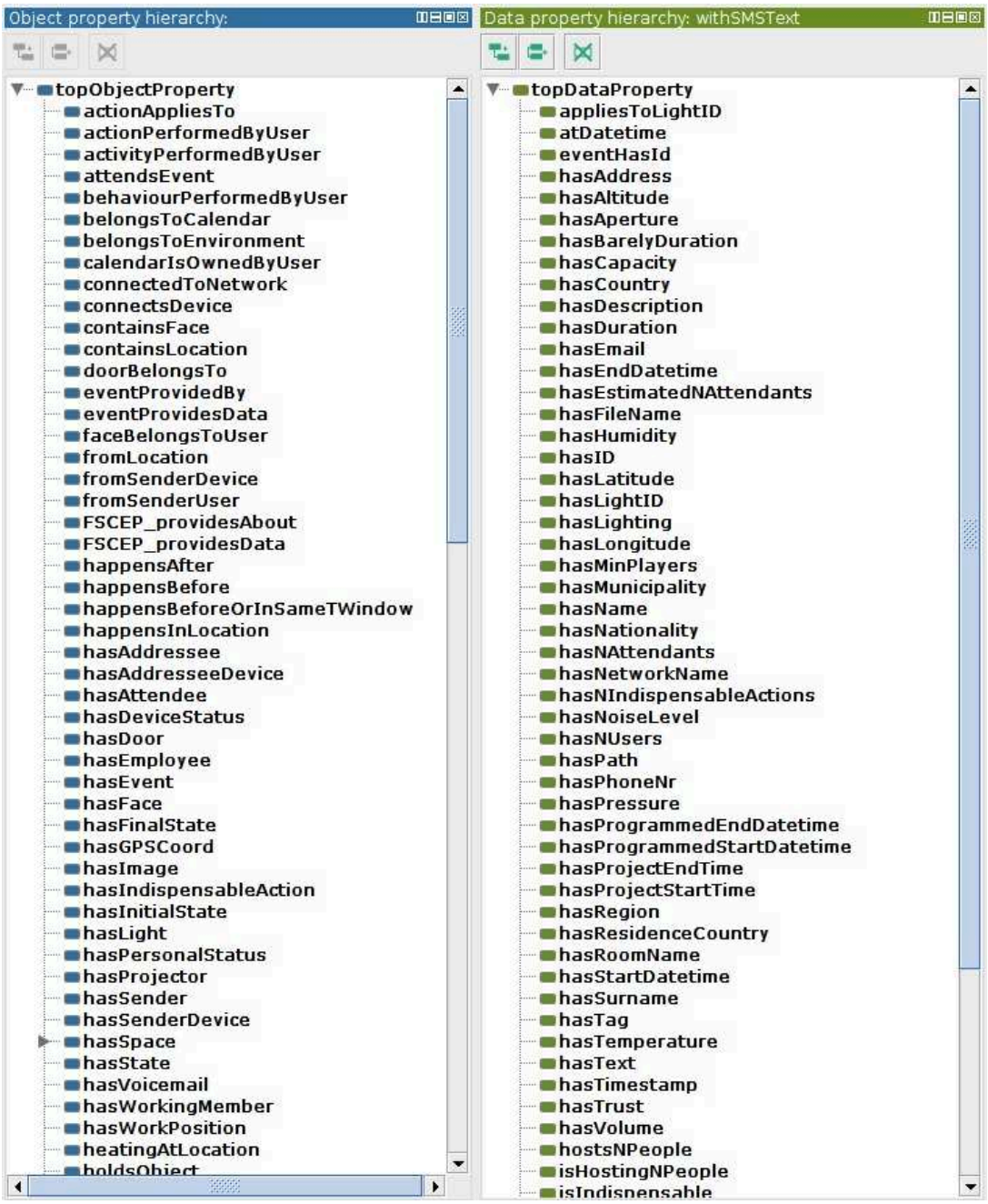
[1] http://users.abo.fi/ndiaz/public/FuzzyHumanBehaviourOntology/FuzzyHumanBehaviourV11.owl



Classes related to devices



Classes related to activities



Excerpt of object and data properties

Appendix E

Publications

1. Ramoly, N., Bouzeghoub, A., and Finance, B. Context-aware planning by refinement for personal robots in smart homes. In *ISR 2016: 47st International Symposium on Robotics; Proceedings of* (2016), VDE, pp. 1–8

2. Jarraya, A., Ramoly, N., Bouzeghoub, A., Arour, K., Borgi, A., and Finance, B. A fuzzy semantic cep model for situation identification in smart homes. In *ECAI 2016: 22nd European Conference on Artificial Intelligence* (2016), vol. 285, IOS Press, pp. 1678–1679

3. Jarraya, A., Ramoly, N., Bouzeghoub, A., Arour, K., Borgi, A., and Finance, B. Fscep: a new model for context perception in smart homes. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (2016), Springer, pp. 465–484

4. Ramoly, N., Vassout, V., Bouzeghoub, A., El Yacoubi, M. A., and Hariz, M. Refining visual activity recognition with semantic reasoning. In *Advanced Information Networking and Applications (AINA), 2017 IEEE 31st International Conference on* (2017), IEEE, pp. 720–727

5. Sfar, H., Bouzeghoub, A., Ramoly, N., and Boudy, J. Agacy monitoring: a hybrid model for activity recognition and uncertainty handling. In *European Semantic Web Conference* (2017), Springer, pp. 254–269

6. Sfar, H., Bouzeghoub, A., Ramoly, N., and Boudy, J. A novel hybrid model for activity recognition. In *Modeling Decisions for Artificial Intelligence* (2017), Springer, pp. 170–182

7. Sfar, H., Ramoly, N., Bouzeghoub, A., and Finance, B. Caredas: Context and activity recognition enabling detection of anomalous situation. In *Conference on Artificial Intelligence in Medicine in Europe* (2017), Springer, pp. 24–36

8. Ramoly, N., Bouzeghoub, A., and Finance, B. A framework for service robots in smart home: an efficient solution for domestic healthcare. In *Journées d'Etude sur la TéléSANté, 6ème edition* (2017)

9. Sfar, H., Bouzeghoub, A., and Ramoly, N. Activity recognition for anomalous situation detection. In *Journées d'Etude sur la TéléSANté, 6ème edition* (2017)

10. Ramoly, N., Sfar, H., Bouzeghoub, A., and Finance, B. Leaf: Using semantic based experience to prevent task failures. In *Field and Service Robotics* (2017), Springer, pp. 681–697

11. Ramoly, N., Bouzeghoub, A., and Finance, B. A causal multi-armed bandit approach for domestic robots' failure avoidance. In *International Conference on Neural Information Processing* (2017), Springer, pp. 90–99

12. Under review: Ramoly, N., Bouzeghoub, A., and Finance, B. A framework for service robots in smart home: an efficient solution for domestic healthcare. In *IRBM - Innovation and Research in BioMedical Engineering*

Appendix F

Résumé en Français

Introduction

L'association de robots personnels et d'intelligences ambiantes est une nouvelle voie pour l'aide à domicile. En effet, en s'appuyant sur les appareils intelligents présents dans l'environnement, les robots pourraient fournir un service de haute qualité. Cependant, même si plusieurs travaux vont dans cette direction, il y a encore de nombreux verrous. C'est pour cela que dans cette thèse, nous cherchons à résoudre les problématiques de perception, de cognition et d'action rencontrées par les robots dans des environnements intelligents, et ainsi assurer la qualité de service de ces derniers.

Problématique

Les problématiques rencontrées par les robots dans les environnements intelligents sont variées, plusieurs projets de recherche, tel que Robot-Era [22] ou CompanionAble [67] ont couvert ces problèmes. Néanmoins, de nombreux verrous ne sont pas résolus. Dans cette thèse, nous avons décidé de nous focaliser sur deux classes de problèmes : la gestion du contexte et la planification de tâche.

La gestion du contexte recouvre la perception et la cognition. Elle consiste en l'acquisition de la connaissance du contexte à partir des différents capteurs, puis à l'utilisation de cette connaissance pour le raisonnement et la prise de décision. L'acquisition du contexte, ou perception, est facilitée par l'utilisation des capteurs à la fois du robot et de l'environnement intelligent. En revanche, une telle variété de capteurs est une source de conflits, de défauts et d'hétérogénéité. Cela peut, en effet, engendrer des données imprécises, erronées, périmées, en contradiction ou incomplètes. L'incertitude est donc un problème majeur. En plus de cela, les données fournies sont hétérogènes. En conséquence, comment le robot peut-il acquérir et comprendre le contexte, l'incertitude et l'hétérogénéité des données ? Une fois la connaissance du contexte acquise, le robot entre dans la phase de cognition pour raisonner et prendre une décision, si besoin est. Le robot doit en particulier réagir à certaines situations qui sont à risques ou inconfortables pour l'utilisateur. Ce dernier doit comprendre et identifier si la situation est anormale, et si c'est le cas, intervenir pour résoudre cette situation. Cependant, les problèmes d'incertitude et d'hétérogénéité persistent. D'où la problématique : comment le robot peut détecter des situations anormales, à l'exécution et malgré l'incertitude et l'hétérogénéité

des données, afin de prendre une décision en conséquence ? Dans l'état de l'art, les techniques proposées recouvrent souvent l'acquisition et l'utilisation du contexte. On y trouve deux grandes classes de solutions [190, 8], celles basées sur l'apprentissage, et celles basées sur une spécification par un expert. Les méthodes d'apprentissage travaillent sur des données de plus bas niveau et supportent souvent plusieurs dimensions d'incertitude. De leur côté, les méthodes par spécification permettent une compréhension plus complète et un raisonnement poussé. Cependant, aucune de ces méthodes n'est capable de faire face à nos problèmes. Des méthodes hybrides ont été explorées [45, 151], même si elles parviennent à de meilleurs résultats, elles ne permettent pas non plus de couvrir tous les problèmes, en particulier concernant l'incertitude.

Pour atteindre un objectif, le robot peut calculer une séquence d'actions à réaliser. Un tel processus se situe dans la phase d'action et peut utiliser de la cognition. Pour construire une séquence d'actions, un planificateur de tâches peut être utilisé. Un tel outil se base sur une connaissance préétablie. Cependant, comme l'environnement dans lequel le robot va travailler n'est pas connu à priori, il est irréaliste de considérer cette connaissance comme exhaustive. De plus, pendant que le robot réalise son plan, des changements peuvent avoir lieu dans l'environnement. Ces deux faits sont problématiques car ils engendrent des échecs d'actions, ce qui n'est pas acceptable. Mais comment le robot peut-il comprendre les causes d'échecs pour ensuite les éviter proactivement ? Par ailleurs, dans un environnement ambiant, il est important de considérer les capteurs et actionneurs. En effet, le planificateur utilise directement ou indirectement des capteurs pour observer le contexte, mais toutes les données de contexte ne sont pas utiles, causant une utilisation inutile des capteurs. Les actionneurs peuvent intervenir dans le plan, mais ne sont pas toujours disponible ou en état de fonctionner, ce qui peut être problématique. Comment le planificateur de tâche peut prendre en compte efficacement l'environnement intelligent ? La plupart des méthodes de l'état de l'art n'abordent pas ce problème [41]. Pour la gestion des échecs, la solution la plus commune consiste à replanifier quand un échec est rencontré [185]. Cette solution est peu acceptable car le robot subit tout de même les échecs, ce qui coûte du temps et de l'énergie. Cependant, peu d'approche explore la possibilité d'éviter les échecs proactivement [86].

Contributions

Dans cette thèse, nous avons proposé plusieurs contributions couvrant les problèmes de perception, de cognition et d'action. Nos propositions s'appuient à la fois sur l'apprentissage et le raisonnement.

Tout d'abord, afin d'acquérir la connaissance du contexte, malgré l'incertitude et l'hétérogénéité des données, nous avons proposé un outil d'acquisition de contexte qui enrichit et floute les événements fournis par différents capteurs. Cela est permis par l'utilisation de CEP, d'ontologies et de le logique floue. Notre approche permet à notre approche de gérer différents types d'événements et quatre dimensions d'incertitude, là où les méthodes classiques n'en abordent que deux ou trois.

Pour la détection des situations anormales, et par la suite la décision, nous avons étudié une méthode hybride basée sur les Réseaux Logiques de Markov et une ontologie de contexte. Cette approche utilise une sémantique de haut niveau permettant une compréhension complexe des situations. De plus, l'incertitude des données est prise en compte via des poids et le modèle fonctionne

malgré une connaissance partielle. Ces caractéristiques permettent de détecter les situations anormales avec une haute précision.

Afin d'agir sur dans l'environnement ambiant en faisant face aux échecs, nous avons proposé une nouvelle approche de planification. Au lieu de planifier et d'exécuter le plan séparément, notre planificateur, basé sur HTN, crée et exécute le plan à la volée, ce qui lui permet d'être réactif aux changements de contexte. Pour cela, il s'appuie sur une notion de *statut* permettant de contrôler l'exécution et l'utilisation des actionneurs. Par ailleurs, notre planificateur est capable de déterminer quelles données sont nécessaires pour calculer le plan, ce qui lui permet d'utiliser seulement la partie de la connaissance de contexte qui est importante.

Cependant, ce planificateur requiert une connaissance préalable qui n'est pas exhaustive. Nous avons proposé une méthode d'apprentissage par renforcement et par expérience qui identifie les causes d'échecs et enrichit la connaissance du planificateur. Notre approche prend en compte les retours de l'utilisateur et modélise le problème comme un problème du bandit manchot. De plus, il utilise l'induction causale pour identifier de possible nouvelles causes d'échec Avec cette connaissance des causes d'échec, le plan peut être généré en évitant les tâches qui ont un fort risque d'échec, permettant d'éviter ces derniers de manière proactive.

Finalement, toutes ces contributions peuvent être regroupées pour former un framework global, FAIRIE. Ce dernier peut être spécifié et configuré de différentes manières afin de correspondre à divers besoins, robots et environnements intelligents.

Expérimentation

Chacune de nos contributions a été implémentées, testées et validées au travers de simulations et/ou de tests à l'aide d'un petit robot humanoïde et d'une plateforme intelligente. Afin de simuler l'environnement intelligent et le robot, nous avons utilisé Freedomotic, un framework pour la conception d'espaces intelligents et qui peut être aisément amélioré via des plug-ins. Cela nous a permis de simuler de nombreuses situations ainsi que l'incertitude, qui sont difficiles à contrôler dans des tests réels. Nous avons cependant également réalisé des tests au sein de la plateforme HadapTIC, située à Télécom SudParis, dans ETOILE. Cette plateforme comprend une salle modulaire et des dizaines de capteurs, tel que des capteurs de mouvement, détecteurs d'ouverture, détecteurs de fumée et thermomètres. Nous avons utilisé le robot Nao, un petit robot humanoïde qui peut facilement interagir avec l'utilisateur. Ces expérimentations nous ont permis de valider la pertinence de chacune de nos contributions et d'obtenir des résultats prometteurs pour FAIRIE.

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Intégration Contextuelle de Données Hétérogènes dans un Environnement Ambiant Ouvert et Opportuniste : Application aux Robots Humanoïdes

Mots clés : Robotique Personnelle, Intelligence Ambiante, Raisonnement, Apprentissage, Ontologies

Résumé : L'association de robots personnels et d'intelligences ambiantes est une nouvelle voie pour l'aide à domicile. Grâce aux appareils intelligents de l'environnement, les robots pourraient fournir un service de haute qualité. Cependant, des verrous existent pour la perception, la cognition et l'action.
En effet, une telle association cause des problèmes de variétés, qualités et conflits, engendrant des données hétérogènes et incertaines. Cela complique la perception du contexte et la cognition, i.e. le raisonnement et la prise de décision. La connaissance du contexte est utilisée par le robot pour effectuer des actions. Cependant, il se peut qu'il échoue, à cause de changements de contexte ou par manque de connaissance. Ce qui annule ou retarde son plan.
La littérature aborde ces sujets, mais n'offre aucune solution viable et complète. Face à ces verrous, nous avons proposé des contributions, autour à la fois du raisonnement et de l'apprentissage. Nous avons d'abord conçu un outil d'acquisition de contexte qui gère et modélise l'incertitude. Puis, nous avons proposé une technique de détection de situations anormales à partir de données incertaines. Ensuite, un planificateur dynamique, qui considère les changements de contexte, a été proposé. Enfin, nous avons développé une méthode d'apprentissage par renforcement et expérience pour éviter proactivement les échecs. Toutes nos contributions ont été implémentées et validées via simulation ou à l'aide d'un robot dans une plateforme d'espaces intelligents.

Title : Contextual Integration of Heterogeneous Data in an Open and Opportunistic Smart Environment: Application to Humanoid Robots

Keywords : Personal Robotics, Ambient Intelligence, Reasoning, Learning, Ontologies

Abstract : Personal robots associated with ambient intelligence are an upcoming solution for domestic care. In fact, helped with devices dispatched in the environment, robots could provide a better care to users. However, such robots are encountering challenges of perception, cognition and action.
In fact, such an association brings issues of variety, data quality and conflicts, leading to the heterogeneity and uncertainty of data. These are challenges for both perception, i.e. context acquisition, and cognition, i.e. reasoning and decision making. With the knowledge of the context, the robot can intervene through actions. However, it may encounter task failures due to a lack of knowledge or context changes. This causes the robot to cancel or delay its agenda.
While the literature addresses those topics, it fails to provide complete solutions. In this thesis, we proposed contributions, exploring both reasoning and learning approaches, to cover the whole spectrum of problems. First, we designed novel context acquisition tool that supports and models uncertainty of data. Secondly, we proposed a cognition technique that detects anomalous situation over uncertain data and takes a decision in accordance. Then, we proposed a dynamic planner that takes into consideration the last context changes. Finally, we designed an experience-based reinforcement learning approach to proactively avoid failures.
All our contributions were implemented and validated through simulations and/or with a small robot in a smart home platform.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France